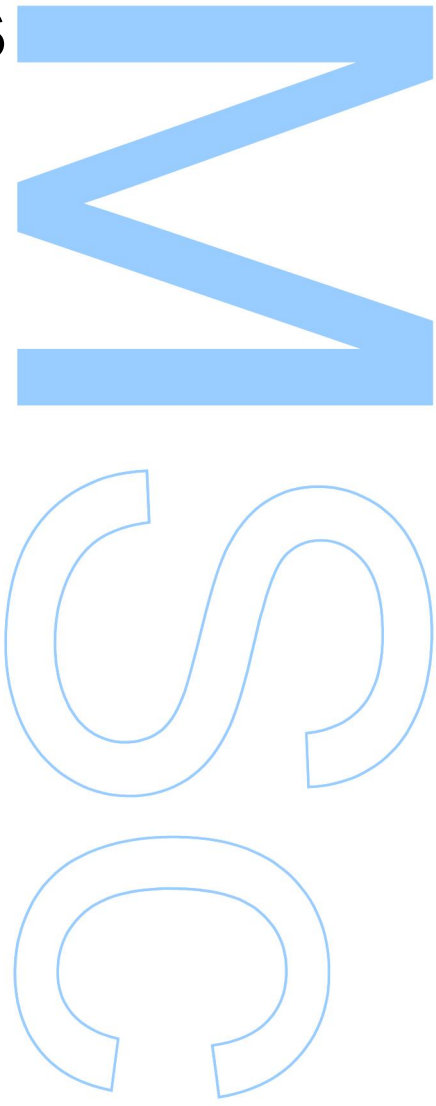
Análise de Sinal Fisiológico usando Métodos não Lineares

João Manuel Monteiro dos Santos
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2015

Orientador
Luís Filipe Coelho Antunes, Professor Associado
Faculdade de Ciências da Universidade do Porto

Coorientador
Cristina Maria Nogueira da Costa Santos, Professora Auxiliar
Faculdade de Medicina da Universidade do Porto



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Para os meus Pais e Avós

Agradecimentos

Gostaria de agradecer, de uma forma geral, a todos os que de certa forma me acompanharam e ajudaram a que este trabalho pudesse ser realizado.

Em particular, e em primeiro lugar, quero agradecer ao Professor Luís Antunes por me ter proposto este trabalho e por me ter dado a oportunidade de trabalhar com ele. A sua simpatia, disponibilidade, apoio e métodos de trabalho proporcionaram a que todas as etapas de concretização desta dissertação fossem realizadas com enorme satisfação, alegria e grande aprendizagem.

Em segundo lugar, o meu agradecimento vai para a Professora Cristina Santos, que sempre se mostrou disponível para ajudar na análise de resultados, propor sugestões e ter proporcionado a oportunidade de assistir às suas aulas.

As pessoas a quem eu mais devo, por me terem proporcionado uma educação de excelência e oportunidade de ingressar no ensino superior são os meus pais. A eles devo tudo o que sou hoje.

Quero agradecer ao meu irmão Rui, essencialmente por ter ajudado os pais nas tarefas domésticas durante o período em que estive menos disponível.

À Rita, um muito obrigado pela empatia demonstrada, paciência que teve comigo, e por me ter ajudado a corrigir erros presentes na dissertação.

Quero também agradecer à Teresa Henriques pela ajuda que forneceu na realização da tese, através não só dos seus trabalhos, como também pela disponibilidade demonstrada.

Por fim, uma palavra de agradecimento ao Luís Maia, pelo tempo disponibilizado na instalação de *software* necessário para a realização da tese.

Resumo

Numa sociedade onde a procura de novas técnicas de avaliação de sinais fisiológicos se tornou evidente, o bem-estar de um bebé à nascença é sem dúvida uma prioridade. Por isso, e por forma a tentar diminuir a mortalidade e morbilidade de um feto, a análise de cardiotocogramas através do estudo de traçados de batimento cardíaco fetal é cada vez mais uma realidade.

Nesta tese, tivemos como objetivo analisar os traçados de batimento cardíaco fetal, procurando características biológicas que estejam relacionadas de certa forma com a compressão de ficheiros. Para isso, foi comparado o desempenho de três compressores diferentes (bzip2, gzip e paq8l) e avaliado se a utilização da técnica de *wavelets* traz algum benefício no estudo de traçados de batimento cardíaco fetal. Por fim, foi utilizado o algoritmo *CompLearn* e a respetiva árvore de classificação no processo de clustering dos traçados.

A menos da capacidade de compressão, os três diferentes compressores apresentaram comportamentos semelhantes na análise de traçados obtidos entre as semanas 24 e 40 da gravidez. Notámos, em geral, uma maior compressão dos traçados de fetos considerados patológicos, um comportamento similar entre géneros com exceção da semana 27 para os traçados de fetos considerados normais, e uma diferença de comportamento nos patológicos após as 36 semanas de gravidez. Observámos um crescimento estável da taxa de compressão nos traçados de fetos com peso normal à nascença ao contrário daqueles que estão abaixo do percentil 3, onde se nota um crescimento instável.

A técnica *wavelet*, a partir de um conjunto de dados pré-categorizado em Normal, Suspeito e Patológico, através do pH do cordão umbilical à nascença, não trouxe grandes benefícios. Porém, resultados positivos foram obtidos na predição de prematuridade usando a *wavelet daubechies*.

Para avaliarmos o CompLearn foram feitos dois testes. Dividimos o conjunto de dados pelo pH do cordão umbilical e através da variabilidade curta dos traçados. Verificámos

não haver relação entre o pH e a compressão, ao contrário da variabilidade curta, onde traçados com variabilidade curta inferior a 30 e superior a 70 foram separados com sucesso pelo algoritmo e a árvore de classificação.

Abstract

In a society where the search for new evaluation techniques of physiological signals has become a reality, the well being of a baby is without a doubt a priority. Therefore, and in a way to decrease the mortality and morbidity of a baby born, the cardiotocograms' analyses through the study of heart rate frequency tracings is increasingly something to take in account.

In this thesis, we wanted to analyse the fetal heart rate frequency tracings, searching for biological characteristics that are related with data compression. For that, we compared the performances of three different compressors: bzip2, gzip and paq8l. It was also analyzed if the usage wavelet trasforms would be beneficial for the study of tracings of fetus heart rate compared to normal data compression. At last, we used the CompLearn algorithm and its classification tree in the process of clustering the tracings organized either by the pH of the umbilical cord at birth or the short term variability (STV) of the tracings.

Despite of the compression power difference between the three tested compressors, all of them performed quite similarly in the analyes of traces obtained between week 24 and 40 of gestation. Also, we noticed a slightly better compression of traces considered pathologic, a similar behaviour was noted between genders with the exception for week 27 for the normal tracings, and a different behaviour for pathologic tracings after week 36 of pregnancy. A stable growth throughout gestation time was observed in the compression ratio of fetus tracings whose weight at birth was considered normal, however an unstable growth is noticeable for those below weight, namely those under percenile 3.

The wavelet technique was not very helpful using a dataset divided in Normal, Suspect and Pathologic tracings by pH. However, positive results were achieved predicting prematurity of fetuses using the daubechies wavelet.

To evaluate CompLearn two different tests were done. One in which the dataset was

divided by pH of the umbilical cord and the other by short term variability (STV). We verified that there was no relation between compression used in CompLearn and pH. On the other hand, the algorithm successfully clustered tracings with low STV and high STV.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Ao longo da história que o ser humano, por necessidade e não só, se interessou em perceber como o seu organismo funciona. A capacidade de resposta às necessidades de mudança a que o meio nos obriga torna-nos seres complexos compostos por interações dos sistemas fisiológicos cujos comportamentos, muitas vezes, não são previsíveis.

Pelo que se conhece hoje, quantificar a complexidade tornou-se importante para entender as características e relações subjacentes e foi aí que Plesk e Greenhalgh [3] conjeturaram que as componentes da saúde que pertencem aos domínios do simples, complexo e caótico se baseiam no diagrama de Stacy (Figura 1.1). As componentes de medicina onde existe grande concordância e certeza dizem-se pertencer ao domínio do simples e são associadas a comportamentos facilmente predizíveis. No domínio do complexo, estamos perante comportamentos e interações dinâmicas onde o conhecimento é baseado através de múltiplas aproximações que não são facilmente calculadas, gerando níveis médios de certeza e concordância entre os especialistas. Por último, no domínio do caótico, nenhuma informação é visível através do comportamento e das interações fisiológicas, implicando pouca certeza e concordância.

A cardiotocografia é, por todo o mundo, um método de monitorização do batimento cardíaco fetal e contrações uterinas durante a gravidez (anteparto) e durante o parto (intraparto). No entanto, a capacidade de previsão do bem-estar fetal deste método é ainda controverso [4]. Daqui, entende-se que o batimento cardíaco fetal é tido como pertencer aos domínios do complexo e caótico.

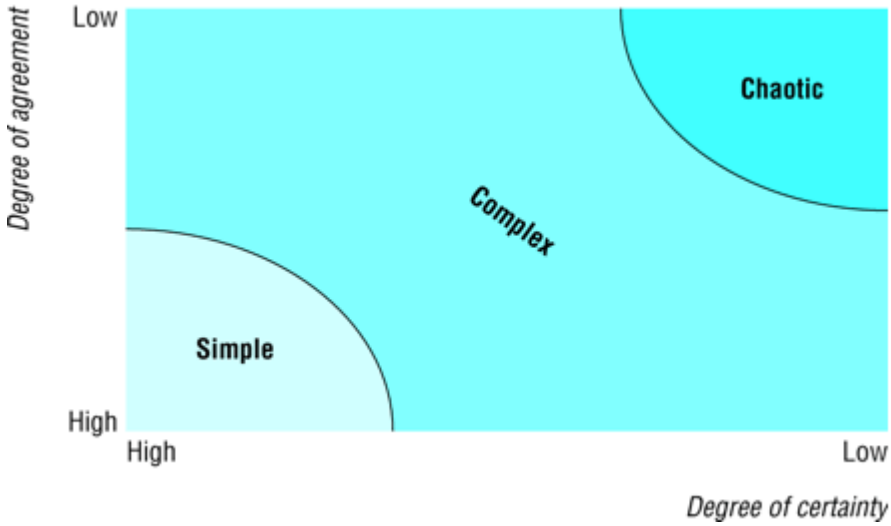


Figura 1.1: Diagrama de certeza-concordância proposto por Stacy.

A análise computacional de cardiotocogramas fornece parâmetros quantitativos que muito dificilmente são percetíveis ao olho humano. O Sisporto é um programa desenvolvido nas últimas duas décadas que tem como função uma análise automatizada dos traçados não só em tempo real como também armazena os dados de forma a que possam ser processados posteriormente. Os conjuntos de dados trabalhados nesta dissertação provieram do Sisporto. Resumidamente, e apenas para introduzir conceitos relevantes para esta tese, a análise dos traçados é feita da seguinte forma: é considerado um ponto de variabilidade curta (STV) anormal quando a diferença entre dois sinais adjacentes é menor que 1 bpm (linhas vermelhas na parte superior da Figura 1.2). A média do STV e a percentagem de pontos com STV anormal são calculados e é possível a sua visualização na parte inferior da Figura 1.2. Quando a diferença entre o máximo e o mínimo, num intervalo de 60 segundos centrado no ponto sem acelerações ou desacelerações, não excede os 5 bpm, é considerado um ponto de variabildade longa (LTV) anormal. As médias de STV e LTV e a percentagem de pontos com STV anormal e LTV anormal são calculados e é possível a sua visualização na parte inferior da Figura 1.2 [5].

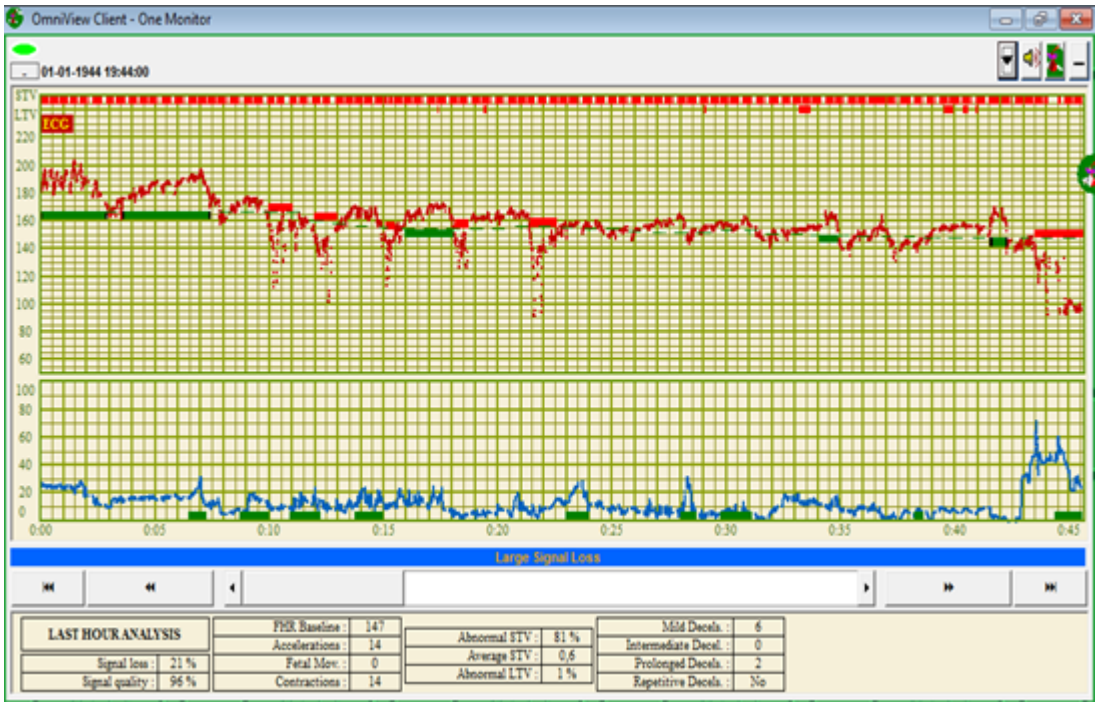


Figura 1.2: Análise de um cardiotocograma pelo Sisporto.

Na definição de estado patológico são tidas duas abordagens: uma usando o pH do cordão umbilical à nascença e outra usando o índice Apgar. O pH do cordão umbilical é um indicador de acidemia fetal e geralmente varia entre 7,05 e 7,4 sendo indicador de bem-estar quando superior a 7,2. O índice Apgar é um sistema de avaliação feito ao primeiro e quinto minutos após o parto onde cinco sinais vitais são avaliados numa escala de 0 a 2, *Appearence* (cor da pele), *Pulse* (frequência cardíaca), *Grimace* (resposta a estímulos), *Activity* (tónus muscular) e *Respiration* (respiração ou choro). Nesta dissertação, o Apgar usado foi ao quinto minuto.

Em 2005, um novo método de clustering usando compressão foi introduzido por Cilibrasi e Vitányi [6] e obteve bons resultados em áreas distintas como música, análise de tráfego na internet, genómica, etc [5].

É o objetivo desta dissertação procurar relações entre as características dos traçados de batimento cardíaco fetal e a compressão. Definimos como taxa de compressão a razão entre o tamanho do ficheiro comprimido e o ficheiro original. Foi também definido rácio de compressão (CR) o inverso da taxa de compressão, isto é, a proporção entre o tamanho do ficheiro original e a respetiva versão comprimida.

1.1 Estrutura da tese

No capítulo 2 serão abordados os conceitos primordiais intrínsecos à compressão de ficheiros, mencionando conceitos de Teoria de Informação, máquinas de Turing e complexidade de Kolmogorov. Será aqui definida a Distância de Compressão Normalizada, conceito importante na descrição do algoritmo CompLearn. Toda esta secção foi baseada no trabalho de Rudi Cilibrasi [1].

No capítulo 3 serão dadas a conhecer as *wavelets*, uma técnica recente desenvolvida com base nas transformadas de Fourier. Estes dois capítulos representam assim o estado da arte.

A análise dos resultados obtidos será feita no capítulo 4, e será dividida em três secções. Em 4.1 serão comparados os resultados dos compressores gzip, bzip2 e paq8l na compressão de traçados e respetivas relações com o desenvolvimento do feto e diferenças entre géneros, tempo de gestação e peso da criança à nascença. No subcapítulo 4.2, irá ser avaliado se a utilização das transformadas *wavelets* traz benefícios na análise de batimento cardíaco fetal e por fim, em 4.3, irá ser comparado o algoritmo CompLearn (e respetiva árvore de classificação) com o pH do cordão umbilical do bebé à nascença e a variabilidade curta do traçado durante o cardiotocograma.

Capítulo 2

CompLearn

Há mais de 30 anos que se desenvolve software com o objetivo de comprimir dados e novos modelos para quase todo o tipo de ficheiros têm aparecido. Até há bem pouco tempo, o interesse nesta tecnologia era direcionado para a diminuição de espaço em disco ou custos de transmissão de dados, mas, ultimamente, uma nova abordagem tem surgido. Aqui, pretendemos abordar este novo conceito que visa usar compressores no sentido de nos ajudarem no processo de *clustering*, que neste trabalho são traçados de batimento cardíaco fetal.

Enquanto grande parte das pessoas está familiarizada com o conceito de compressão de ficheiros, uma propriedade menos conhecida dita que a combinação de dois ou mais ficheiros concatenados, produzindo um ficheiro muito maior antes da compressão, muitas vezes gera melhores rácios de compressão comparada com a compressão individual desses mesmos ficheiros. Esta propriedade é válida para programas como o *tar* ou *pkzip*. O conceito chave subjacente é que se dois ficheiros são muito semelhantes, então o tamanho da versão comprimida desses ficheiros aglomerados será bem menor que a soma dos ficheiros comprimidos separadamente. Por outro lado, se os ficheiros não tiverem nada em comum, este método não trará vantagem nenhuma e o seu resultado será semelhante à soma das versões comprimidas.

Esta ideia bastante intuitiva tem tido resultados surpreendentes, mesmo usando compressores muito simples. No exemplo da Figura 2.1, foi retirado ADN mitocondrial de mamíferos e testado este método. Primeiro foi construída uma matriz, chamada a matriz de distâncias, que calcula o quão semelhante cada par de ficheiros é. Esta distância resulta da comparação dos ficheiros comprimidos mencionada atrás. Depois foi construída uma árvore de acordo com essas distâncias. Os resultados obtidos foram

excecionais (ver [1]).

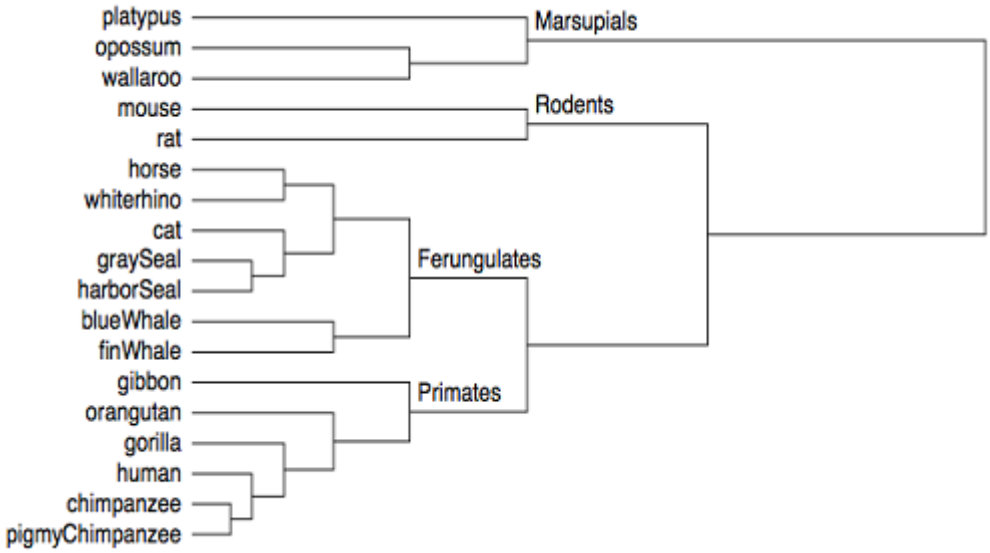


Figura 2.1: Árvore filogenética construída usando ADN mitocondrial de mamíferos [1].

Este procedimento é bastante simples e pode ser usado em diversas áreas. Com base nestas ideias, foi desenvolvido o *CompLearn*. Este software pode ser usado para classificação ou *clustering*. O primeiro requer exemplos introduzidos pelo especialista. *Clustering* não requer nenhuma informação à priori e refere-se à organização dos objetos em grupos. Aqui, é usado um cluster hierárquico chamado *quartet method* que procura a árvore que mais se ajusta ao problema, onde objetos semelhantes estarão próximos, ao contrário daqueles que menos semelhanças apresentam. Este método é não-determinístico e tenta encontrar uma solução num razoável intervalo de tempo, sacrificando alguma eficácia. Como faz uso de números "aleatórios", os resultados obtidos sofrem algumas variações para o mesmo conjunto de dados.

Na próxima secção abordaremos alguma da terminologia e teoria por trás deste método. Na secção 2.2 serão dadas noções de máquinas de Turing e complexidade de Kolmogorov e na secção 3 introduziremos a distância de compressão normalizada (NCD).

2.1 Definições e Introdução Técnica

Esta secção tem o objetivo de familiarizar o leitor acerca de conceitos e notação relevantes para o trabalho. Para informação mais detalhada, o leitor deve ver [1, 7, 8].

2.1.1 Finito e Infinito

Podemos dividir o domínio dos objetos matemáticos em duas grandes categorias: os objetos finitos e o infinitos. Os primeiros são aqueles que podem ser delimitados enquanto os infinitos são sempre maiores do que qualquer possível fronteira. Por exemplo, o conjunto dos divisores de um certo número é finito, mas o conjunto de todos os números primos é infinito, embora seja possível, em teoria, escrever um programa que consiga gerar todos esses números, supondo que se tem tempo e memória ilimitada. A ideia de que conjuntos embora infinitos possam ser gerados por programas finitos é importante.

2.1.2 *Strings* e Linguagens

Um alfabeto é simplesmente um conjunto de símbolos usados para formar uma linguagem. Uma *string* é uma lista ordenada de 0 ou mais símbolos de um alfabeto pré-definido, que neste trabalho é constituído apenas pelos símbolos 0 e 1, pois podemos codificar qualquer alfabeto num alfabeto binário, da mesma forma que um byte é codificado em bits.

Tem-se que $\Sigma = \{0, 1\}$ é o alfabeto a ser usado e Σ^* constitui o espaço de todas as possibilidades de *strings*, incluindo a *string* vazia. Para uma dada *string* x, $|x|$ representa o seu comprimento, em bits.

2.1.3 Códigos Instantâneos e Univocamente Decifráveis

Diz-se que uma *string* y é prefixo de x se pudermos escrever $x = yz$, para um z diferente da *string* vazia. Um conjunto diz-se livre de prefixos (ou instantâneo) se nenhum elemento do conjunto for prefixo de outro. Os elementos de um conjunto livre de prefixos são chamados de palavras código. Os códigos instantâneos têm a vantagem de serem facilmente decifráveis porque os limites de cada palavra código estão bem definidos, não dando hipóteses a ambiguidades.

Consideremos o conjunto dos naturais e a sua representação binária. De acordo com a Figura 2.2, notam-se dois números de comprimento 1, 4 de comprimento 2, e por aí fora. No entanto, existem menos palavras código livres de prefixo para cada comprimento. Assintoticamente, existem menos palavras código livres de prefixo de tamanho n do que 2^n palavras fonte de tamanho n.

x	$\|x\|$	n
0	1	1
1	1	2
00	2	3
01	2	4
10	2	5
11	2	6
000	3	7
001	3	8
010	3	9
011	3	10
100	3	11

Figura 2.2: Exemplo dos primeiros onze naturais, respetivas representações binárias e tamanho da *string* [1].

Esta observação é válida para códigos de prefixo arbitrários. A quantificação desta intuição para um conjunto numerável e um código instantâneo arbitrário leva-nos a uma relação designada como Desigualdade de Kraft:

Lema *Seja $n_1, n_2, \ldots, n_k$ uma sequência finita ou infinita de naturais. Existe um código instantâneo para esta sequência como comprimentos do seu código binário se e só se*

$$\sum_k 2^{-n_k} \leq 1. \tag{2.1}$$

Queremos codificar elementos de um determinado conjunto de forma a que aquando da descodificação estes sejam reconstruidos univocamente. A estes códigos damos o nome de univocamente decifráveis. Todo o código livre de prefixos é univocamente decifrável, embora o contrário nem sempre aconteça. Isto é importante porque significa que se pode substituir qualquer código univocamente decifrável por um instantâneo sem que o conjunto dos tamanhos das palavras código se altere.

Um código univocamente decifrável diz-se completo se a adição de qualquer nova palavra código ao conjunto de palavras código resulta num código que já não é univocamente decifrável. Isto tem uma relação muito próxima com a desigualdade de Kraft pois acontece quando a igualdade se verifica. Sejam $l_1, l_2, \ldots$ palavras código de um código univocamente decifrável completo. Seja $q_x = 2^{-l_x}$. Por definição, temos que $\sum_x q_x = 1$. Então q_x pode ser pensado como sendo a função massa de probabilidade correspondente a uma distribuição Q para uma variável aleatória X. Diz-se que Q é a distribuição correspondente para $l_1, l_2, \ldots$. Desta forma, cada código completo univocamente decifrável é mapeado numa única distribuição de probabilidade. É claro que isto é uma correspondência meramente formal, pois podemos codificar elementos de um conjunto usando um código que corresponde a uma distribuição q, onde o resultado é distribuído segundo $p \neq q$. Mas se o conjunto for distribuído segundo p, então o código correspondente a p é, em média, o código que atinge a compressão ótima. Em particular, qualquer função massa de probabilidade p está relacionada com o código de prefixo de Shannon-Fano, de modo que o número esperado de bits por palavra código transmitida é o mais baixo possível para qualquer código de prefixo. Este código codifica a palavra x numa palavra código de tamanho $l_x = \lceil \log \frac{1}{p(x)} \rceil$, onde $P(X = x) = p(x)$, de tal forma que o tamanho esperado da palavra código transmitida seja $\sum_x p(x) \log \frac{1}{p(x)} = H(X)$, a entropia de Shannon, que será referida mais à frente.

2.2 Máquinas de Turing

Esta secção serve como preparação para posteriormente ser introduzido o conceito de complexidade de Kolmogorov. Informalmente, a complexidade de Kolmogorov de uma *string* é o programa mais curto que a imprime e depois pára. Para tornar esta definição mais precisa, usam-se as máquinas Turing universais, que são uma representação matemática abstrata de um computador. São uma generalização e uma simplificação de máquinas computacionais determinísticas. Uma máquina de Turing recebe como parâmetro de entrada uma cadeia de caracteres (que pode ser visto como um programa) e seguindo um certo conjunto de regras imprime o resultado desse programa. Tal como existem linguagens de programação universais, i.e. linguagens onde é possível escrever um compilador para qualquer outra linguagem, também existem máquinas de Turing Universais, que conseguem simular qualquer outra máquina.

De uma forma geral, uma máquina de Turing consiste em:

- uma fita dividida em células, cada uma preenchida por um símbolo de um

alfabeto finito (que contém um símbolo que denota espaço em branco). É assumido que a fita é extensível em ambos os lados tanto quanto necessário e que as células não escritas são constituídas pelo símbolo branco;

- um apontador que se movimenta para a esquerda e direita (um passo) e que consegue ler e escrever. No estado inicial situa-se no símbolo não branco mais à esquerda;
- um registador de estados que armazena o estado da máquina, incluindo o estado inicial e final. O número de estados é finito;
- uma função de transição que irá indicar para onde mover o apontador, que símbolo escrever e em que estado se irá encontrar, com base no símbolo de leitura e estado atual. Quando a função devolver o estado final, a máquina pára.

Como interessam apenas os programas que retornam uma *string* finita, isto é programas que não entram em ciclo, dizemos que a função de transição apenas está definida para programas que não resultem nestes loops.

Para além disso, olhamos apenas para máquinas de Turing cujos programas formam códigos livre de prefixos, ou seja, nenhuma palavra do código é prefixo de outra. Isto é feito para evitarmos o problema da subaditividade na Complexidade de Kolmogorov.

2.3 Complexidade de Kolmogorov

Formalmente, definimos Complexidade de Kolmogorov K, como uma função unária que mapeia a *string* x num inteiro e está implicitamente condicionada a uma Máquina de Turing Φ:

$$K_\Phi(x) = \begin{cases} \min\{|p| : & \Phi(p) = x\}, \\ \infty, & \text{se p não existe.} \end{cases}$$

Por outras palavras, e como já foi dito, K é o tamanho do programa mais pequeno que imprime x e pára. Esta relação é identificada como $K_\Phi(x)$. Na prática, queremos usar uma máquina de Turing o mais genérica possível. Exige-se que o conjunto de programas forme um conjunto livre de prefixo. Não existe perda de generalidade porque qualquer máquina de Turing universal consegue simular outra, segundo o Teorema da Invariância.

2.3.1 Complexidade de Kolmogorov condicionada

O conceito de complexidade de Kolmogorov condicionada é um pouco mais difícil de perceber. Esta forma da função K, definida como $K(z|y)$ espera não um argumento mas sim dois. Podemos defini-la como o tamanho, em bits, do menor programa que imprime z dada uma codificação. Podemos entender y como o input inicial da fita. A ideia é que se y tiver muita informação acerca de z, então $K(z|y) \ll K(z)$, mas se y e z não estiverem relacionados, teremos $K(z|y) \approx K(z)$.

2.3.2 Aleatoriedade e Compressão

O objetivo de Kolmogorov com este trabalho era obter uma definição formal de sequência aleatória e observou que algumas sequências binárias poderiam ser comprimidas segundo um algoritmo. Como se constata, K fornece um meio para caracterizar sequências aleatórias. Contrariamente ao que se possa pensar, sequências aleatórias não são simplesmente sequências onde padrões não sejam discerníveis. Existem regularidades estatísticas que podem ser observadas e provadas, mas a dificuldade encontra-se simplesmente em expressá-las.

Podemos definir aleatoriedade da seguinte forma: diz-se que uma *string* é k-aleatória se e só se $K(x) > |x| - k$. Esta relação expressa a ideia que *strings* aleatórias não são compressíveis.

Olhemos para o número π. Se nos focarmos nas casas decimais e estendermos o número infinitamente, a *string* parece ser aleatória, mas sendo possível escrever um programa que imprima os algarismos deste número, esta ideia de que o π é aleatório cai por terra. Esta produção de um programa (finito) que gera uma sequência infinita implica uma certa regularidade nos dados com grande probabilidade.

Argumentos mostram que o número de programas passíveis de serem altamente comprimidos é bastante reduzido. Em particular, o rácio de programas comprimíveis em apenas k bits não é mais do que 2^{-k}. Para uma *string* de tamanho m temos 2^m possibilidades. Se considerarmos codificações de tamanho m de uma fonte de *string* de tamanho $n > m$, então temos no máximo 2^m *strings* diferentes codificadas de um total de 2^n. Então, o rácio de *strings* passíveis de compressão por $n - m$ bits é no máximo 2^{m-n} do total.

2.3.3 Universalidade de K

Já foi referido como máquinas de Turing universais conseguem simular outras máquinas usando programas de simulação finitos. Falando do comportamento assintótico de K, estas simulações são similares a menos de uma constante aditiva. Por exemplo, se tivermos x^n para representar os primeiros n dígitos de π, então $K(x^n) = O(\log n)$ independentemente da máquina de Turing universal que se escolheu. Isto acontece porque é possível calcular qualquer número de dígitos de π usando um programa de tamanho fixo que lê o número de dígitos a imprimir. O comprimento deste input não pode ser codificado em menos de $\log n$ bits.

Isto implica que todas as variações de K sejam, de uma certa forma, equivalentes porque qualquer duas variações de K, dadas duas máquinas de Turing universais, não divergirão mais de que uma constante que não depende da *string* mas sim das máquinas de Turing. É esta universalidade que credibiliza a ideia que K pode ser usado como medida absoluta de informação de um objeto.

Teorema *Teorema da Invariância*

Para qualquer máquina de Turing M e string binária x, e para algum $c > 0$, existe uma máquina U, dita máquina Universal, tal que $K_U(x) \leq K_M(x) + c$, onde c depende de M e não de x.

Prova *Para $K_M(x) = \infty$, a desigualdade é trivialmente verdadeira. Consideremos uma máquina universal U tal que $U_{1^{*i}0p} = M_{i_p}$, onde M_i é a i-ésima máquina na enumeração das máquinas. Seja M a n-ésima máquina, isto é, $M_n = M$. Então*

$$K_M(x) = \min_{M_p = x} |p| \tag{2.2}$$

e

$$K_U(x) = \min_{U_{1^{*n}0p} = x} |1^{*n}0p| = \min_{U_{1^{*n}0p} = x} |p| + n + 1 = K_M(x) + n + 1. \tag{2.3}$$

Ou seja, o limite superior da complexidade expressa em U é $K_M(x) + n + 1$ e eventualmente existe um p' tal que $U_{p'} = x$ e $|p'| < |p| + n + 1$. Assim, $K_U(x) \leq K_M(x) + n + 1$ onde tomando $c = n + 1$ depende apenas de M.

2.3.4 K vs Probabilidade Clássica

Nesta secção usaremos o exemplo da moeda para compararmos a complexidade de Kolmogorov e a probabilidade clássica. Primeiro vamos supor que temos uma moeda

equilibrada e que serão feitas N experiências. Naturalmente, podemos dizer que a sequência gerada segue uma distribuição de Bernoulli onde a variável aleatória X toma valores 0 ou 1 com probabilidade

$$P(X = 0) = \frac{1}{2} = P(X = 1), \tag{2.4}$$

onde P representa a probabilidade de um certo evento ocorrer. Neste caso, podemos representar os resultados através de um simples bit onde para N experiências teremos N bits, independentemente do resultado.

Imaginemos agora uma moeda não equilibrada, dada por

$$P(X = 0) = \frac{1}{8}, \qquad P(X = 1) = \frac{7}{8}. \tag{2.5}$$

Aqui, poderá ser feita uma codificação diferente de modo a que em média se usem menos bits para representar uma sequência de resultados. Vamos assumir que N é par e vamos dividir a sequência resultante em pares, codificando em 1 os pares cujos ambos os algarismos são 1, e em 0 caso contrário. No sentido de não se perder informação, quando a codificação é 0 é também adicionado o par, ficando a palavra código com 3 bits. Em média, o número de bits necessários para representar a sequência é dado por l:

$$l = \frac{1}{2}(\frac{49}{64} + \frac{15 \cdot 3}{64}) = \frac{94}{128} \tag{2.6}$$

Este resultado pode ainda ser melhorado segundo a entropia de Shannon H(X), usando blocos maiores ou codificações mais eficientes:

$$H(X) = \sum -P(X = i) \log_2 P(X = i) \tag{2.7}$$

$$= -\frac{1}{8} \cdot \log_2(\frac{1}{8}) - \frac{7}{8} \cdot \log_2(\frac{7}{8}) < \frac{94}{128} \tag{2.8}$$

Pelo teorema de Shannon, este é o tamanho mais curto do código, em média.

As leis de probabilidade clássica funcionam para um número considerável de experiências, mas não para uma sequência em particular. Medir a incerteza desta forma não parece ser a melhor abordagem. Com a complexidade de Kolmogorov, isto muda de figura, pois dizemos que uma *string* é aleatória se não for compressível. Voltando ao exemplo da moeda equilibrada, embora a entropia seja de 1 bit por cada lançamento da moeda, não podemos dizer com certeza que uma dada sequência não possa ser substancialmente comprimida. Isso é apenas verdade com grande probabilidade.

2.3.5 Incomputabilidade da Complexidade de Kolmogorov

Embora em teoria o conceito de complexidade de Kolmogorov seja objetivo, na prática não funciona tão bem pelo simples facto de não ser computável.

Uma outra abordagem poderá ser feita utilizando compressores. A ideia de compressão de dados provém do problema de eficientemente codificarmos uma certa sequência no menor número de bits possível de forma a que a reconstrução da sequência original seja feita sem qualquer perda de informação. Esta é a ideia básica de um compressor, sendo alguns dos mais usados incluem o *gzip*, *bzip2*, o *PPM* ou compressores da série *paq*:

gzip é um compressor do tipo Lempel-Ziv com uma janela de 32KB. É muito simples e rápido.

bzip2 é mais recente e usa o algoritmo blocksort. Providencia boa compressão e uma janela expandida de 900KB permitindo um maior leque de padrões a serem detetados. É também razoavelmente rápido.

PPM é sigla para *Prediction by Partial Matching*. É parte de uma nova geração de compressores poderosos que usam uma mistura de modelos estatísticos organizados por árvores, árvores de sufixos ou arrays de sufixos. No entanto é muito lento e consome muita memória.

Compressores *paq* são concorrentes ao *PPM*. Embora sendo também muito lentos, recorrem a redes neuronais.

Estes compressores são usados para aproximar, por cima, o valor da função K. É de notar que os compressores mencionados operam numa base de bytes e não bits, fazendo com que detalhes subtis passem despercebidos na análise de *strings* pequenas. No sentido de ultrapassar este problema, o *CompLearn* suporta a ideia de um compressor virtual (sugerido por Steven Rooij), isto é, em vez do resultado ser uma forma codificada, simplesmente acumula o número de bits necessários para codificar o resultado usando um hipotético codificador aritmético ou de entropia. Isto liberta-nos da restrição da codificação de base em bytes e elimina a necessidade de arredondamento para um número inteiro de bits. Em vez disso, podemos retornar valores reais. Isto torna-se muito útil principalmente quando se está a analisar *strings* com menos de 100 bytes.

2.4 Distância de Compressão Normalizada

Nesta secção irá ser abordada a Distância de Compressão Normalizada (NCD) como também a Distância de Informação Normalizada (NID). Como já foi dito NCD é uma medida de similaridade baseada em compressores, enquanto NID é simplesmente uma instanciação de NCD usando o compressor teórico ideal (e incomputável) de Kolmogorov.

2.4.1 Métrica de Semelhança

Em matemática, diferentes conceitos de distância são usados dependendo do contexto, daí a necessidade de definir uma métrica. Neste caso, usaremos o "grau de similaridade"presente em áreas de reconhecimento de padrões.

Métrica: Seja Ω um conjunto não vazio e $\Re^+$ o conjunto dos números reais não negativos. A métrica Ω é uma função $D : \Omega \times \Omega \to \Re^+$ que satisfaz as seguintes relações:

- $D(x, y) = 0 \Leftrightarrow x = y$;
- $D(x, y) = D(y, x)$ (simetria);
- $D(x, y) \leq D(x, z) + D(z, y)$ (desigualdade triangular).

O valor de $D(x, y)$ é a distância entre x e y. A métrica mais usada é a Euclidiana, que suporta a noção de distância que usamos no dia a dia. Neste caso estamos interessados numa métrica de similaridade entre os objetos.

Densidade: Na definição de uma classe de distâncias admissíveis (e não necessariamente distâncias métricas) é usual excluir situações irrealistas como quando $f(x, y) = \frac{1}{2}$ para todo o $x \neq y$. Isto é feito para restringir o número de objetos em torno de uma certa distância de um outro objeto. Por isso, consideremos então o seguinte:

Definição *Seja $\Omega = \Sigma^*$, onde Σ é um alfabeto finito não vazio e Σ^* o conjunto de strings finitas desse alfabeto. Como qualquer alfabeto pode ser codificado em binário, então é usado $\Sigma = \{0, 1\}$. Uma função $D : \Omega \times \Omega \to \Re^+$ é uma distância admissível se para cada par de objetos $x, y \in \Omega$, a distância $D(x, y)$ satisfaz a condição de densidade*

(uma versão da Desigualdade de Kraft):

$$\sum_{y} 2^{-D(x,y)} \leq 1, \tag{2.9}$$

Se D for uma distância admissível, então para todo x, o conjunto $\{D(x,y) : y \in \{0,1\}^*\}$ é o conjunto das distâncias de um código de prefixos, pois satisfaz a Desigualdade de Kraft (equação 2.1). Reciprocamente, se uma distância é o conjunto comprimento de um código instantâneo, então satisfaz a Desigualdade de Kraft.

Normalização: Objetos grandes (como *strings* longas) que diferem pouco são intuitivamente mais próximos, do que objetos pequenos que diferem na mesma quantidade. Desta forma, a diferença absoluta entre dois objetos não proporciona bons resultados de similaridade, ao contrário da diferença relativa.

Definição *Um compressor é um codificador sem perdas que mapeia Ω em $\{0,1\}^*$ de tal forma que o código resultante seja instantâneo. "Sem perdas"significa que o descodificador é capaz de reaver a mensagem original sem que algum dado se tenha perdido. Define-se compressor como uma função $C : \Omega \to N$, onde N é o conjunto de inteiros não negativos. Assim, a versão comprimida de x é o comprimento $C(x)$. Apenas serão considerados compressores que satisfaçam $C(x) \leq |x| + O(\log|x|)$, onde a parcela logarítmica representa a necessidade do objeto comprimido ser uma palavra código livre de prefixos.*

Definição *Seja D uma distância admissível. Podemos definir $D^+(x) = \max\{D(x,z) : C(z) \leq C(x)\}$ e $D^+(x,y) = \max\{D^+(x), D^+(y)\}$.*

É de notar que tal como $D(x,y) = D(y,x)$, $D^+(x,y) = D^+(y,x)$.

Definição *Seja D uma distância admissível. A distância admissível normalizada, também chamada distância de similaridade, $d(x,y)$ é definida como*

$$d(x,y) = \frac{D(x,y)}{D^+(x,y)}. \tag{2.10}$$

Das definições, segue que a distância admissível normalizada é uma função $d : \Omega \times \Omega \to [0,1]$, simétrica.

Lema *Para todo $x \in \Omega$ e constante $e \in [0,1]$, a distância admissível normalizada satisfaz a condição da densidade*

$$|\{y : d(x,y) \leq e, C(y) \leq C(x)\}| < 2^{eD^+(x)+1} \tag{2.11}$$

A uma distância normalizada $d(x, y)$ é chamada distância de similaridade porque retorna uma similaridade relativa de acordo com a distância (sendo 0 a máxima similarida e 1 quando os objetos são completamente distintos).

2.4.2 Compressor Normal

Definição *Um compressor C diz-se normal se satisfizer, até um termo aditivo* $O(\log n)$*, com n sendo o comprimento máximo binário de um elemento de* Ω *envolvido na (des)igualdade em questão, as seguintes condições:*

- *Idempotência:* $C(xx) = C(x)$ *e* $C(\lambda) = 0$*, para a string vazia* λ*;*
- *Monotonia:* $C(xy) \geq C(x)$*;*
- *Simetria: C(xy) = C(yx);*
- *Distributividade:* $C(xy) + C(z) \leq C(xz) + C(yz)$

Compressores da família Lempel-Ziv, como o *gzip*, e da família PPM não são precisamente simétricos. O ficheiro inicial x pode ter certas regularidades na qual o compressor se adapta. Após passar a fronteira para y, tem de se reajustar para as novas regularidades de y. Este processo poderá causar algumas imprecisões na simetria que desaparece à medida que o comprimento de x e y aumenta. Para além desta grande família, existe uma outra cujos compressores são baseados em codificação por blocos, como é o caso do *bzip2* que consideram todas as rotações do bloco de entrada na obtenção da versão comprimida.

Definição *Seja* $C(y|x)$ *o número de bits de informação de y relativo a x definido por*

$$C(y|x) = C(xy) - C(x) \tag{2.12}$$

$C(y|x)$ pode ser visto como o número de bits excessivos na versão comprimida de xy comparado com a versão comprimida de x, e é chamado de quantidade de informação comprimida condicional.

Na definição de compressor o algoritmo de descompressão não está incluído (ao contrário do que acontece na complexidade de Kolmogorov, onde esta é dada por definição), mas a construção de um é trivial: Dada a versão comprimida de x, em $C(x)$ bits, podemos, embora seja um processo ineficiente, usar um compressor em todos os candidatos z

até que apareça um tal que $C(z_0) = C(x)$, resultando em $z_0 = x$. Pela unicidade de descompressão, temos que $C(y|x)$ é o número extra de bits exigidos para descrever y após a descrição de x. É intuitivo que a informação comprimida condicional $C(y|x)$ satisfaz a desigualdade triangular

$$C(y|x) \leq C(y|z) + C(z|x). \tag{2.13}$$

Lema *Um compressor normal satisfaz a propriedade de subaditividade:* $C(xy) \leq C(x) + C(y)$.

Esta propriedade é claramente exigida para uma compressão viável porque um compressor deve poder usar informação presente em x para comprimir y. Tal como referido anteriormente, alguma imprecisão poderá surgir na fronteira entre x e y, mas será insignificante para x e y suficientemente grandes.

2.4.3 Distância de Informação Normalizada

Em [7] a distância de informação $E(x, y)$ foi introduzida e definida como o tamanho do programa mais curto que dado input x imprime y. Os autores mostraram que a menos de uma quantidade logarítmica, $E(x, y) = \max\{K(x|y), K(y|x)\}$. Mostraram também que $E(x, y)$ é uma métrica, embora com algumas violações das desigualdades mencionadas. Em [8], a versão normalizada de $E(x, y)$, chamada distância de informação normalizada (NID) foi definida como sendo:

$$NID(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{2.14}$$

Esta medida é também uma métrica. Se dois objetos forem similares de acordo com uma determinada característica descrita por uma distância admissível normalizada em particular, então também serão similares no que toca a NID. É de notar que diferentes pares de objetos poderão ter características dominantes diferentes. Ainda assim, similaridades dominantes como essas serão detetadas pela NID. Infelizmente, como esta métrica é baseada na complexidade de Kolmogorov, não é computável pelo que uma aproximação terá de ser feita. No que toca ao denominador da equação 2.14, dado compressor C, a transformação é trivial pois é simplesmente $\max\{C(x), C(y)\}$. Por outro lado, o numerador é mais difícil. Pode ser reescrito como $\max\{K(x, y) -$

$K(x), K(x,y) - K(y)\}$, com uma certa precisão logarítmica. O termo $K(x,y)$ representa o tamanho do programa mais curto para o par (x,y). Como na prática é mais fácil usar a concatenação xy ou yx, a menos de uma imprecisão logarítmica temos $K(x,y) = K(xy) = K(yx)$. Seguindo a referência de Steven Rooij, podemos aproximar o numerador a $\min\{C(xy), C(yx)\} - \min\{C(x), C(y)\}$. Como compressores baseados em codificação por blocos são simétricos quase por definição, e experiências com compressores baseados em fluxos (PPMZ, gzip) mostram apenas alguns desvios da simetria, a ferramenta CompLearn apenas usa $C(xy)$ em vez de $\min\{C(xy), C(yx)\}$.

O resultado da aproximação de NID usando um compressor C é chamado de distância de compressão normalizada (NCD) (e será formalizada posteriormente). No sentido de preencher a lacuna existente entre a teoria e a prática, a teoria de NCD irá ser desenvolvida baseada em axiomas já mencionados. Será mostrado que NCD é uma métrica de similaridade semi-universal relativo a um compressor normal C.

2.4.4 Distância de Compressão

Aqui vamos definir distância de compressão baseada num compressor normal e indicar que é uma distância admissível. Na aplicação desta abordagem, é necessária uma aproximação partindo de um compressor C do "mundo real"muito menos poderoso. Este compressor aproxima a distância de informação $E(x,y)$, baseada em complexidade de Kolmogorov, pela distância de compressão definida como

$$E_C(x,y) = C(xy) - \min\{C(x), C(y)\}, \tag{2.15}$$

onde $C(xy)$ é o tamanho dos ficheiros concatenados x e y após a compressão.

Lema *Se C é um compressor normal, $E_C(x,y) + O(1)$ é uma distância admissível.*

Lema *Se C é um compressor normal, então $E_C(x,y)$ satisfaz as (des)igualdades métricas até uma precisão logarítmica.*

Lema *Se C é um compressor normal, então $E_C^+ = \max\{C(x), C(y)\}$*

2.4.5 Distância de Compressão Normalizada

A aproximação da distância de informação normalizada (equação 2.14) usando um compressor C, que por outras palavra é a versão normalizada da distância admissível $E_C(x,y)$, é chamada de distância de compressão normalizada ou NCD:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \tag{2.16}$$

O valor de NCD varia entre $[0, 1+\epsilon]$ e representa o quão diferentes dois objetos são. Quanto mais próximo de 0, mais similares são. A presença de ϵ deve-se a pequenas imprecisões, mas para a maioria dos algoritmos de compressão usuais, é improvável encontrar $\epsilon > 0.1$.

Teorema *Se um compressor é normal, então NCD é uma distância admissível normalizada que satisfaz as (des)igualdades métricas, isto é, uma métrica de similaridade.*

Semi-Universalidade: A motivação para o desenvolvimento do NCD foi estudada em [8]. Toda a distância admissível que expressa similaridade de acordo com uma determinada característica e que possa ser computada, é minorada pela NID. Notemos que cada característica dos dados dá origem a uma similaridade e, reciprocamente, cada similaridade pode ser entendida como uma expressão de alguma característica. Infelizmente, a prática de NCD fica aquém desta teoria em pelo menos três aspetos:

- A universalidade de NID é garantida apenas para sequências indefinidamente longas. A partir do momento que se consideram *strings* de comprimento n, é apenas universal no que diz respeito a distâncias admissíveis normalizadas que sejam "simples"de computar, onde "simples"significa que são computáveis por programas de comprimento logarítmico em n.

- A complexidade de Kolmogorov não é computável e, de certa forma, não é possível calcular o quão distante é a NCD da NID.

- Para a aproximação da NCD são usados programas de compressão como o *gzip*, *bzip2*, *PPMD* ou *paq8l*. Enquanto uma melhor compressão de uma *string* irá aproximar melhor o valor da complexidade de Kolmogorov, isto não se verifica com NCD. Devido à sua forma de construção, é teoricamente possível que enquanto todos os objetos de uma fórmula sejam melhor comprimidos, esse progresso não seja uniforme em todos os objetos e assim, o valor de NCD desvia-se de NID.

2.5 Ferramenta CompLearn

Temos como objetivo aplicar o algoritmo NCD do *CompLearn* a ficheiros de batimento cardíaco fetal. A cada par de ficheiros é calculado o NCD construindo assim uma matriz, chamada de matriz de distâncias. As entradas dessa matriz serão as distâncias entre cada par de traçados, isto é, o quão similares eles são. Sendo esta uma forma mais compacta de reunir informação, estamos em condições de aplicar algoritmos de *clustering*. Como já referido, o *CompLearn* utiliza o método dos quartetos exatos que produz uma árvore ternária em que as folhas representam os ficheiros e os ramos representam similaridades entre os mesmos. Em cada passo, o algoritmo tende a aperfeiçoar a árvore modificando as sub-árvores. A cada árvore resultante está associado um valor de $S(t)$ que indica o quão bem representada está a matriz pela árvore.

Infelizmente, sendo o método da construção da árvore um problema *NP-hard*, não é escalável, impedindo na prática de ser usado para números de objetos desejáveis.

Capítulo 3

Técnicas de compressão com perdas

Com o avanço tecnológico, tem existido um crescimento exponencial dos dados digitais obtidos através de sinais fisiológicos medidos em exames como o eletrocardiograma (ECG), eletroencefalograma (EEG), cardiotocograma (CTG), etc. Isto fez com que um armazenamento e transmissão eficientes destes dados fossem uma prioridade [9]. Por um lado, temos uma grande quantidade de dados a armazenar o que leva a um grande volume de informação. Por outro, técnicas de transmissão, através de canais de comunicação, permitem a peritos aceder a essa informação com uma boa relação custo-benefício. Por isso, é reconhecida a necessidade de boas técnicas de compressão de dados provenientes dos sinais fisiológicos, no sentido de reduzir a quantidade de bits de informação necessários para armazenar e transmitir o sinal [10]. Para além disso, notou-se uma necessidade de obter mais informação do sinal do que aquela existente à primeira vista (domínio temporal), informação que muitas vezes se encontra na análise de frequência [2]. Cientistas e engenheiros que trabalham com dados obtidos do mundo real sabem que os sinais não existem sem ruído. Em condições ideais, o ruído pode decrescer para níveis onde poderá ser desprezível, enquanto que o sinal aumenta para níveis significativos. Infelizmente, o sinal é corrompido por ruído mais vezes do que o desejado e tem de ser removido para que o sinal seja analisado. A primeira questão que se coloca é: Será que esta remoção do ruído deverá ser feita no domínio do sinal original (espaço-tempo) ou num outro domínio? Neste último caso, várias técnicas têm sido estudadas, através das transformadas de Fourier, ou mais recentemente, das transformadas *wavelet*, entre outras [11].

A compressão de sinal pode ser *lossy* ou *lossless*. Usando um algoritmo de compressão *lossless*, a informação comprimida pode ser usada para recriar na totalidade a informação original e nenhum dado é perdido no processo. Este tipo de compressão é

também conhecido por *entropy coding*. Este nome provém do facto do sinal comprimido ser geralmente mais aleatório que o original, pois padrões são removidos aquando da compressão. Embora a compressão *lossless* seja útil na reconstrução exata, geralmente não atinge o rácio de compressão esperado. Na compressão *lossy*, o sinal original não pode ser reconstruido na sua totalidade. A razão por trás disto é que muitos dos detalhes do sinal, não sendo importantes para análise, podem ser descartados. O rácio de compressão aqui atingido é normalmente superior, em troca de alguma informação que é perdida. É neste campo que se incluem os métodos mencionados, as transformadas de Fourier e as transformadas *wavelet* [12]. Nesta dissertação será feita uma introdução às Transformadas de Fourier, pela importância que têm, e posteriormente introduziremos as *wavelets*, que são o nosso objeto de estudo.

3.1 Transformada de Fourier

A transformada de Fourier é um dos métodos tradicionais mais conhecidos no processamento de sinal digital com aplicações em análise de frequências, análise de sinal, etc.

Fourier representou aproximações de funções usando combinações de senos e cossenos. O primeiro passo é transformar uma função com domínio temporal numa função de domínio de frequência. O sinal pode então ser analisado porque os coeficientes de Fourier da função transformada representam a contribuição de cada seno e cosseno em cada frequência existente.

Quando os pares ordenados, que representam a função de input, estão igualmente espaçados (como por exemplo, em intervalos de tempo), dizemos que estamos perante uma Transformada de Fourier Discreta (DFT). Sendo este o caso de todos os nossos dados de input, apenas nos vamos debruçar sobre a variante discreta destas transformadas.

Uma sequência de dados $x_0, ..., x_{N-1}$ de tamanho N, é transformada numa sequência de números complexos, de periodicidade N, através da seguinte equação:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i k n}{N}} \tag{3.1}$$

Os valores de X_k são os coeficientes usados na função de aproximação. Por outro lado, a transformada inversa (IDFT) executa o processo inverso, isto é, transforma uma função de domínio de frequência, numa de domínio temporal:

$$X_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k e^{-\frac{2\pi i k n}{N}} \tag{3.2}$$

A DFT tem propriedades de simetria quase iguais às existentes nas Transformadas de Fourier Contínuas. Adicionalmente, a equação (3.2) é de cálculo fácil pois é muito parecida com a equação (3.1).

Podemos ter dois tipos de sinal, estacionário e não-estacionário. No primeiro caso, todas as frequências existentes não se alteram com o tempo (Figura 3.1). No segundo, essas frequências alteram-se (Figura 3.2). Como exemplo deste último, temos o sinal Chirp, demonstrado na Figura 3.3. Aqui, reparamos que embora as funções no domínio temporal sejam diferentes, quando transformadas para o espetro de frequência tornam-se iguais. Levanta-se uma questão: Em que tempo ocorre cada um dos componentes de frequência? As transformadas de Fourier não conseguem responder, apenas nos informam quais componentes de frequência existem no sinal. A informação de tempo e frequência não podem ser vistas ao mesmo tempo (Princípio da Incerteza de Heisenberg).

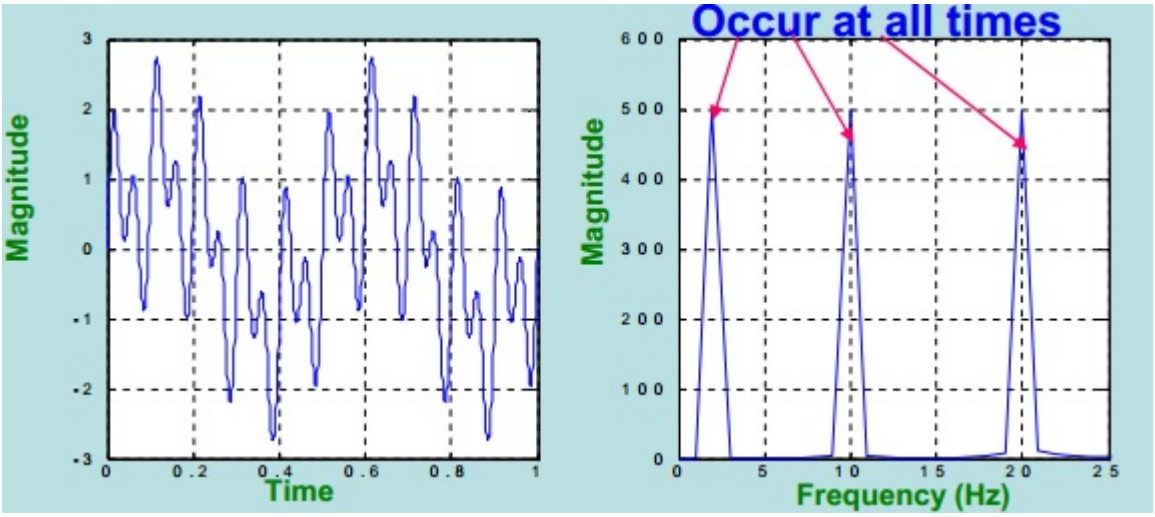


Figura 3.1: Exemplo de sinal estacionário (2Hz + 10Hz + 20Hz).

Muitos dos sinais existentes são não-estacionários, mas temos a necessidade de saber quando é que determinada frequência ocorre. Uma solução encontrada (Dennis Gabor, 1946) foi a Transformada de Fourier de curto termo (STFT). Esta técnica consiste

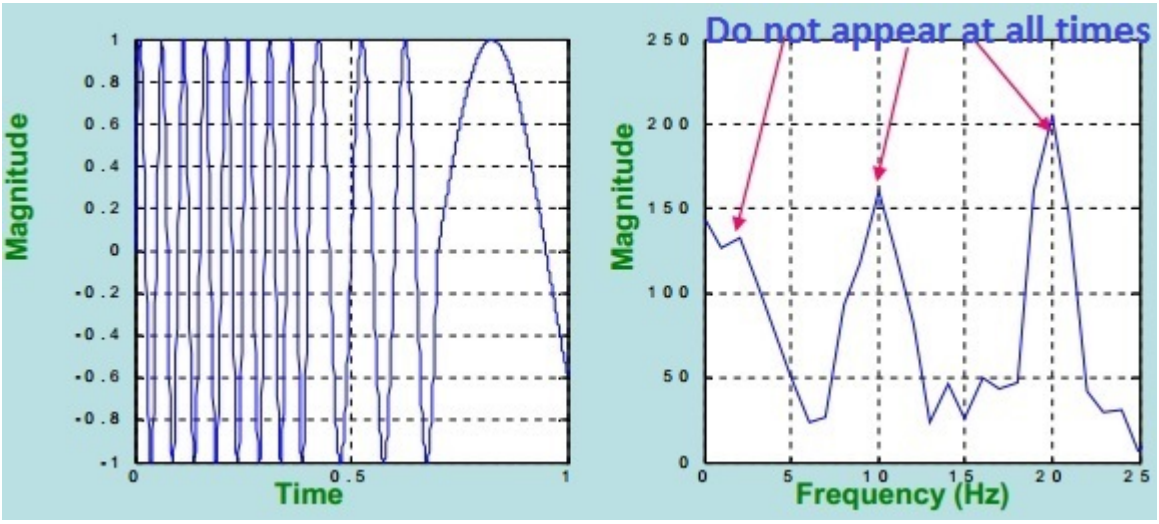


Figura 3.2: Exemplo de sinal não estacionário (2Hz (0.0 a 0.4) + 10Hz (0.4 a 0.7) + 20Hz (0.7 a 1)).

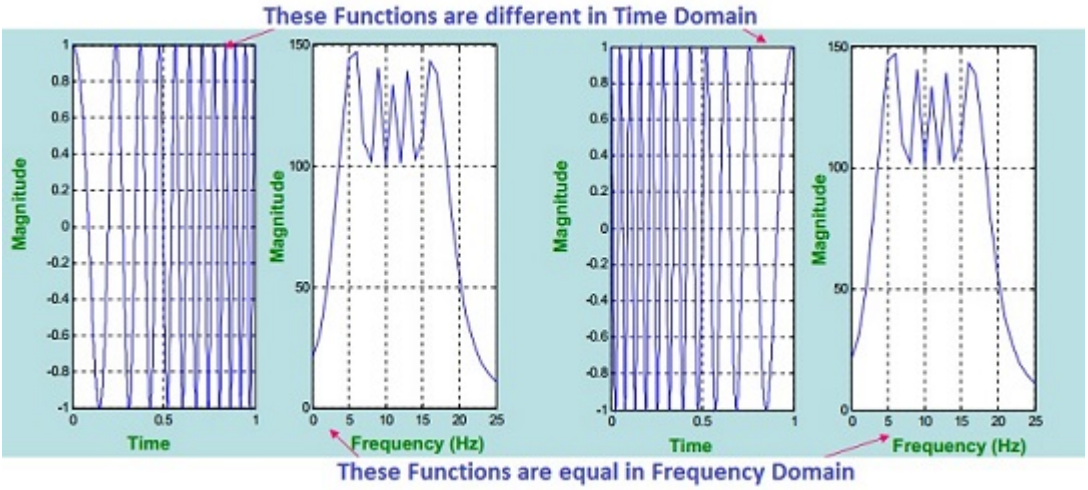


Figura 3.3: Sinais Chirp.

numa análise espectral dependente do tempo: a função é particionada em intervalos, de forma a que o espectro possa ser considerado constante (estacionário) no interior de cada um deles. Uma variação da transformada de Fourier é então aplicada a cada intervalo. Infelizmente esta técnica tem alguns problemas. O intervalo definido mantém-se inalterado durante todo o processo e isso provoca um dilema. Se for escolhido um intervalo pequeno (boa resolução temporal) teremos uma fraca resolução de frequência. Por outro lado, se a janela for demasiado extensa, a resolução temporal será fraca (Figura 3.4).

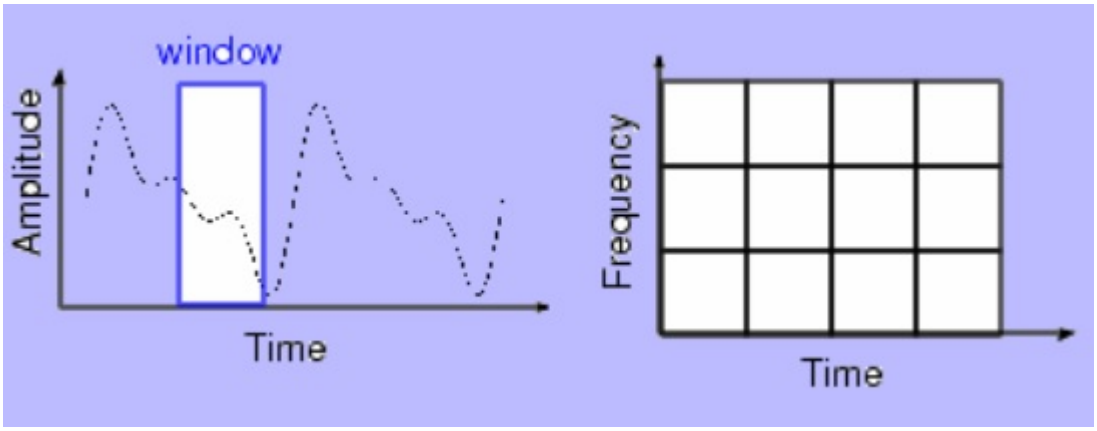


Figura 3.4: Relação intervalo tempo-frequência.

Por norma, é desejada uma abordagem mais flexível onde seja possível variar a dimensão da janela com o objetivo de determinar mais precisamente o tempo ou a frequência [13].

No caso da função ser não-periódica, o somatório das funções seno e cosseno não representa o sinal com grande precisão. Podemos, no entanto, estender artificialmente o sinal de modo a que se torne periódico mas isso iria requerer uma continuidade adicional nas extremidades. A transformada de Fourier por janelas (WFT) é uma solução ao problema da representação de funções não-periódicas. Esta alternativa pode ser usada para obter mais informação do sinal no que toca tanto ao tempo como à frequência. Neste caso, o sinal é cortado em secções e em cada uma delas é analisada a frequência. Quando existem transições brutas no sinal, os dados são manipulados de forma a que essas secções convirjam para zero [14]. Este efeito é conseguido através de uma função peso que tem menos ênfase nas extremidades do que no centro da janela.

Para aproximar uma função por amostras, e aproximar o integral de Fourier pela DFT, requer-se a aplicação de uma matriz cuja ordem é o número de pontos na amostra, n. Como multiplicar uma matriz $n \times n$ por um vetor tem um custo de n^2 operações, o problema torna-se muito dispendioso à medida que a amostra aumenta. Mas, se as amostras estiverem uniformemente espaçadas, então a matriz de Fourier pode ser fatorizada num produto de apenas algumas matrizes e o resultado pode ser aplicado a um vetor num total de operações de ordem $n \log n$. Esta é a chamada transformada rápida de Fourier (FFT) [9, 15, 16].

3.2 Transformada *Wavelet*

Existem várias boas razões para usar *wavelets* na aproximação de funções. Conseguem detetar bem dados com descontinuidades [2]. São simples, práticas, rápidas e facilmente adaptáveis a espaços e frequências não-homogéneos. São extensíveis a grandes dimensões e aplicáveis a vários tipo de problemas tais como estimação de densidade, análise de imagem, processamento de sinal, etc [17]. Esta técnica tem, também, sido muito eficaz na remoção de ruído em sinal, estimando a série real de uma série corrompida por ruído eletrónico ou de eventos externos como: vento, vibrações, variações de temperatura, variações de humidade, etc [18]. Isto é útil para fins de previsão usando diretamente as séries temporais geradas através da decomposição da *wavelet* [19]. Por todas estas razões, as *wavelets* têm sido usadas nas mais diversas áreas como por exemplo: astronomia, acústica, engenharia nuclear, processamento de imagem e sinal, ciência de computadores, neurofisiologia, música, ótica, fractais, previsão de terramotos, radar, visão humana e aplicações matemáticas puras. Um exemplo de sucesso no uso desta técnica foi quando o FBI se deparou com a problemática de ter uma base de dados de impressões digitais com cerca de 30 milhões de entradas, e a crescer. O custo de manutenção começava a ser insuportável. Com a aplicação das *wavelets*, conseguiram atingir um rácio de compressão de 15:1, um resultado bem melhor que a tradicional compressão JPEG [2].

3.2.1 Perspetiva Histórica

O desenvolvimento da análise de *wavelets* conheceu origens diferentes ao longo da história. Muito do trabalho desenvolvido foi feito antes da década de 30 do século passado. Tudo começou com o trabalho desenvolvido por Joseph Fourier (1807) com as suas teorias de análise de frequência, onde afirmou que qualquer função periódica de período 2π é a soma da sua série de Fourier:

$$a_0 + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx) \tag{3.3}$$

Os coeficientes a_0, a_k e b_k são calculados da seguinte forma:

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)\,dx,$$
$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)\,dx, \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)\,dx \tag{3.4}$$

Esta asserção de Fourier foi essencial para a evolução de ideias matemáticas envolvendo funções. Uma dessas noções é a de análise de escala que a partir de uma função (sinal) cria estruturas matemáticas que variam em escala. Esta técnica de análise é menos sensível a ruído porque mede as flutuações médias em diferentes escalas.

A primeira vez que o termo *wavelet* apareceu foi num anexo da tese de A. Haar (1909). Uma propriedade da *wavelet* Haar é ter suporte compacto, isto é, o suporte de uma função é o menor subconjunto fechado do domínio onde a função não é nula. Se esse subconjunto for fechado e limitado dizemos que estamos perante um suporte compacto. Isto significa que fora de um certo intervalo fechado toma valor 0. Infelizmente, *wavelets* Haar não são continuamente diferenciáveis o que limita, em parte, as suas aplicações. Por outro lado, por serem muito simples, são bastante úteis na introdução do conceito *wavelet*.

Nos anos 30, vários grupos independentes investigaram a representação de funções usando funções base de escala variável. Função base é um elemento de um conjunto de funções que, num determinado espaço e através de combinações lineares, são capazes de representar qualquer função contínua. Usando funções de base Haar, o físico Paul Levy conseguiu obter melhores resultados na análise de detalhes em sinal aleatório do que as funções de base de Fourier [20].

Ainda nessa década, Littlewood, Paley e Stein estiveram envolvidos na definição de energia (E) de uma função f, que é dada por:

$$E(f) = \frac{1}{2}\int_0^{2\pi} |f(x)|^2\,dx \tag{3.5}$$

Os resultados obtidos notavam uma distinção entre a energia estar concentrada à volta de alguns pontos ou distribuida ao longo de um intervalo grande. Isto poderia mostrar que a energia poderia nao ser conservada. Estes investigadores descobriram, então, uma função que varia em escala e que conseguia conservar a energia. Isto foi muito importante para David Marr, quando descobriu um algoritmo eficaz no processamento de imagem usando *wavelets*, no início dos anos 80.

Entre os anos 60 e 80, os matemáticos Guido Weiss e Ronald R. Coifman estudaram os elementos mais simples de um espaço funcional, chamados átomos, com o objetivo de encontrar os átomos de uma função em comum, e encontrar regras que permitam reconstruir todos os elementos do espaço funcional usando estes átomos. Na década de 80, Grossman e Morlet, definiram *wavelets* num contexto de física quântica.

Em 1985, Stephane Mallat deu um grande contributo à investigação de processamento de sinal digital. Ele descobriu algumas relações entre filtro de espelho em quadratura, algoritmos de pirâmide e bases *wavelet* ortonormais. Inspirado, em parte, por estes resultados, Y. Meyer construiu os primeiros *wavelets* não triviais. Ao contrário dos *wavelets* Haar, os *wavelets* Meyer são continuamente diferenciáveis, embora não tenham suporte compacto. Uns anos mais tarde, Daubechies pegou no trabalho de Mallat e definiu um conjunto de *wavelets* com funções de base ortonormais que são muito provavelmente uns dos *wavelets* mais elegantes e mais usados nos dias de hoje. [16, 2]

3.2.2 *Wavelet*: Definição

Wavelets são funções que satisfazem uma série de parâmetros matemáticos e são usadas na representação de dados ou outras funções. O termo *wavelet* provém do facto de flutuarem em torno do eixo, integrando-se para zero (as áreas acima do eixo e abaixo são exatamente iguais). *Wavelets* são transformadas multi-resolução, isto é, permitem a análise no tempo e em escala (ao contrário das transformadas de Fourier), e geralmente têm uma forma irregular [21]. Algoritmos *wavelet* processam a informação em escalas (ou resoluções) diferentes. Se olharmos para o sinal através de uma "janela"grande, notamos as características mais amplas. Se, por outro lado, usarmos uma "janela"pequena, notamos os detalhes do sinal. O resultado na análise *wavelet* é, por assim dizer, podermos observar tanto as árvores como as florestas [16].

O procedimento na análise *wavelet* é adotar, em primeiro lugar, uma função *wavelet* protótipo, chamada mãe-*wavelet*. A análise temporal é feita através de uma versão contraída e de alta frequência da mãe-*wavelet*, enquanto que na análise de frequência é usada uma versão dilatada e de baixa frequência. Como o sinal original pode ser representado em termos de expansão *wavelet* (usando coeficientes numa combinação linear das funções *wavelet*), as operações podem ser feitas usando somente os coeficientes [16].

Por uma questão de simplicidade vamos restringir-nos apenas ao espaço de funções

$L^2(R)$, isto é, $f \in L^2(R)$ se e só se

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty \tag{3.6}$$

Por outras palavras, estamos perante o conjunto de funções de domínio real cujo quadrado é integrável [17].

Supondo que a função $f \in L^2(R)$, podemos construir a função $\psi(x)$ tal que [22, 23]:

$$f(x) = \sum_{k \in Z} c_{0,k}\phi_{0,k}(x) + \sum_{j<J} \sum_{k \in Z} d_{j,k}\psi_{j,k}(x), \tag{3.7}$$

tal que

$$c_{0,k} = \int_R f(x)\phi_{0,k}(x) \, dx, \qquad d_{j,k} = \int_R f(x)\psi_{j,k}(x), \tag{3.8}$$

onde J controla a resolução/escala máxima. As funções $\psi_{j,k}(x)$ derivam da função mãe-*wavelet* $\psi(x)$ segundo a relação

$$\psi_{j,k}(x) = \frac{1}{\sqrt{|j|}}\psi(\frac{x-k}{j}). \tag{3.9}$$

Estas funções derivadas, $\psi_{j,k}(x)$, são chamadas de *wavelets* e a função mãe-*wavelet* é escolhida de forma a que o conjunto $\{\psi_{j,k}(x)\}$ forme uma base ortonormal em $L^2(R)$. As funções $\phi_{0,k}(x)$ são derivadas da função $\phi(x)$, conhecida por função pai-*wavelet* ou função de escala, usando a função

$$\phi_{0,k}(x) = \phi(x-k). \tag{3.10}$$

Para um valor de j grande temos uma dilatação do sinal, o que nos leva a observar baixas frequências, estando assim ao alcance de analisar todo o sinal. No caso oposto, para valores baixos de j (baixa resolução), o sinal é comprimido, tomando altas frequências, o que leva à ocupação do mesmo num curto período de tempo [2].

Estamos agora em condições de introduzir a definição de transformada *wavelet* contínua que é dada pela expressão

$$CWT_f^{\psi}(j,k) = \int f(x)\overline{\psi_{j,k}(x)}dx, \tag{3.11}$$

onde, $\overline{\psi_{j,k}(x)}$ é o conjugado complexo de $\psi_{j,k}(x)$.

Vejamos uma ideia geral de como funciona o algoritmo de computação das *wavelets*[2]:

1. A *wavelet* é colocada no início do sinal, com $j = m$ para um valor de m previamente definido. Esta será a forma mais comprimida;
2. Com essa escala, a *wavelet* é multiplicada pelo sinal e integrada pelo seu todo e depois multiplicada pelo inverso da raiz da escala, $\frac{1}{\sqrt{j}}$;
3. Movemos a *wavelet* para o valor $x = k$ e voltamos a calcular o valor da transformada;
4. Repetimos o ponto anterior até que a *wavelet* chegue ao fim do sinal;
5. A escala é aumentada um certo valor suficientemente pequeno e repetimos todo o procedimento feito até aqui. Isto é feito para todo o valor de j;
6. Cada computação para um dado j irá preencher uma linha na matriz tempo-escala;
7. Se todos os valor forem calculados, a matriz será o output.

O valor máximo de escala é definido previamente. Quanto maior for, maior será a exigência computacional.

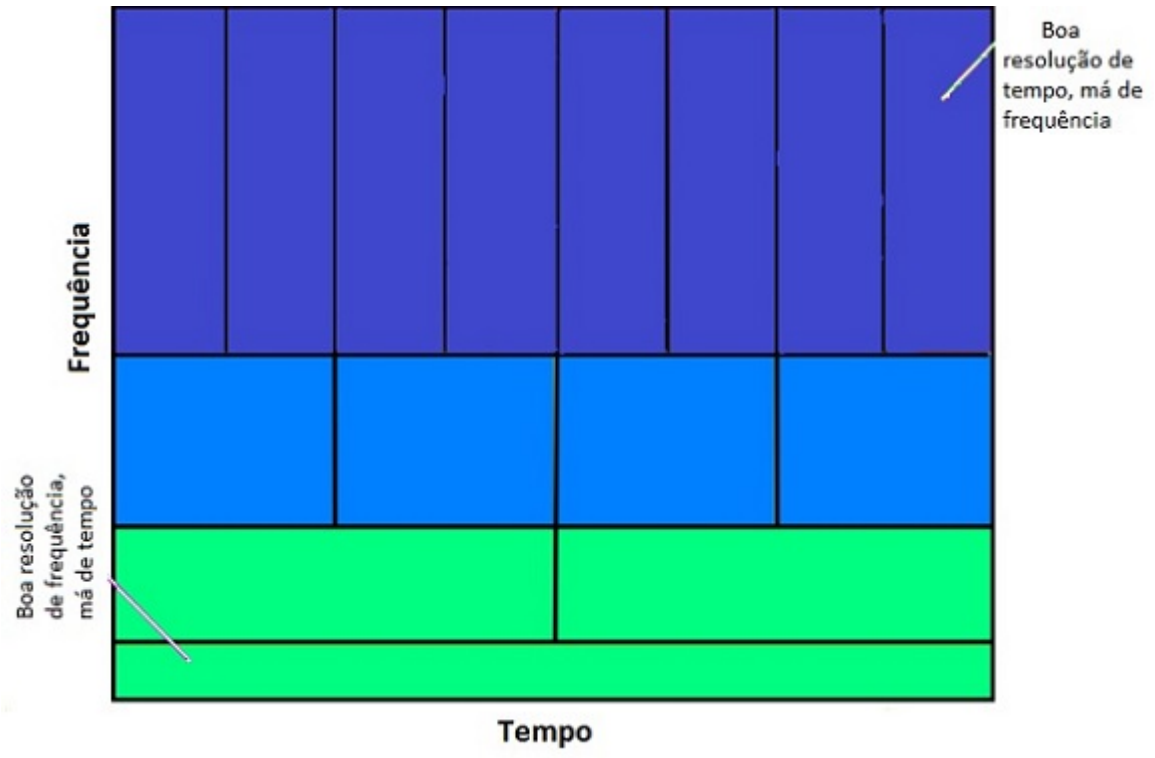


Figura 3.5: Wavelet: Resolução de tempo e frequência.

A normalização por $\frac{1}{\sqrt{|j|}}$ assegura que $\|\psi_{j,k}(x)\|$ seja independente dos valores de j e k. Assumimos que a função mãe-*wavelet* satisfaz a condição de admissibilidade

$$C_\psi = \int_R \frac{|\Psi(\omega)|^2}{|\omega|}\, d\omega < \infty, \tag{3.12}$$

onde

$$\Psi(\omega) = \int_R \psi(x) e^{-ix\omega}\, dx \tag{3.13}$$

é a transformada de Fourier de $\psi(x)$. Esta condição implica que

$$\Psi(0) = \int_R \psi(x)\, dx = 0. \tag{3.14}$$

Por outro lado, se as duas condições seguintes forem satisfeitas, para algum $\alpha > 0$, então garantimos $C_\psi < \infty$.

$$\int_R \psi(x)\, dx = 0, \qquad \int (1 + |x|^\alpha)|\psi(x)|\, dx < \infty. \tag{3.15}$$

Quando a condição de admissibilidade é satisfeita, é então possível encontrar a transformada contínua inversa através da relação conhecida por *resolução de identidade*,

$$f(x) = \frac{1}{C_\psi} \int_{R^2} CWT_f(j,k)\psi_{j,k}(x) \frac{dj\, dk}{j^2}. \tag{3.16}$$

Se restringirmos j a valores reais positivos, o que é mais do que natural, pois pode ser interpretado como o recíproco da frequência, temos que (3.12) se torna

$$C_\psi = \int_0^\infty \frac{|\Psi(\omega)|^2}{\omega}\, d\omega < \infty, \tag{3.17}$$

e a relação resolução de identidade toma a forma

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_0^\infty CWT_f(j,k)\psi_{j,k}(x) \frac{1}{j^2} dj\, dk. \tag{3.18}$$

De seguida, serão enunciadas algumas propriedades das *wavelets* [24]:

(escala e deslocamento) Se $f(x)$ tem uma transformada *wavelet* contínua $CWT_f(j,k)$, então $g(x) = \frac{1}{\sqrt{s}} f(\frac{x}{s})$ tem transformada *wavelet* contínua $CWT_g(j,k) = CWT_f(\frac{j}{s}, \frac{k}{s})$. Por outro lado, se $g(x) = f(x - \beta)$, então g tem transformada *wavelet* contínua $CWT_g(j,k) = CWT_f(j, k - \beta)$.

(conservação de energia) Da equação (3.18),

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{0}^{\infty} |CWT_f(j,k)|^2 \frac{1}{j^2} dj \, dk. \tag{3.19}$$

Tipicamente, *wavelets* de classe m são construídas especialmente de forma a que [23]:

(base ortonormal) o conjunto $\{\psi_{jk}(x)\}$ forme uma base ortonormal para o espaço em consideração;

(regularidade) se $m = 0$, $\psi(x)$ pertence a $L^\infty(R)$; se $m > 1$, $\psi(x)$ e todas as suas derivadas até ordem m pertencem a $L^\infty(R)$;

(localização) $\psi(x)$ e todas as suas funções derivadas até ordem m decresçam rapidamente assim que $x \to \pm\infty$.

(oscilação) $\int_R x^k \psi(x) \, dx = 0$ para $0 \leq k \leq m$.

A ortogonalidade é uma propriedade ideal para a representação de um sinal compacto. Ela garante que os dados não estejam sobre-representados. A propriedade da localização mencionada estende-se também para o domínio da frequência. Tipicamente, *wavelets* estão bem localizados no tempo e frequência. Para isto, *wavelets* normalmente têm um suporte compacto num dos dois domínios (nunca nos dois, obedecendo ao princípio da incerteza) [17].

3.2.3 DFT vs DWT

As transformadas de Fourier e *wavelet* têm não só semelhanças como diferenças. Tanto a FFT como a DWT (vista mais à frente) são operações lineares que geram uma estrutura de dados que contém $\log_2 n$ segmentos de vários comprimentos, geralmente preenchendo e transformando-o em diferentes vetores de comprimento 2^n.

As propriedades matemáticas das matrizes envolvidas nas transformadas também são similares. A matriz da transformada inversa para ambas é a matriz transposta da original. Assim, podem ser vistas como uma rotação no espaço funcional para um domínio diferente. Na FFT, o novo domínio contém funções bases que são senos e cossenos. Na DWT temos funções base mais complicadas, as *wavelets*. Outra semelhança é que ambas as funções base estão localizadas em frequência, fazendo com que instrumentos matemáticos como o cálculo da energia contida num intervalo de frequência e escalogramas sejam úteis na seleção de frequências e cálculo de distribuição de energia.

Uma das diferenças mais relevantes entre as duas transformadas é que as funções seno e cosseno são definidas em todo o domínio, enquanto que as *wavelet* são de suporte compacto, isto é, definidas num intervalo (localizadas no tempo). Esta propriedade, juntamente com a localização da frequência das *wavelets*, faz com que funções e operadores *wavelet*, quando transformados para o domínio *wavelet*, facilitem a deteção de características importantes através de coeficientes de valor absoluto alto (*sparseness*). Isto resulta num número variado de aplicações úteis como compressão de dados, deteção de propriedades em imagens, e remoção de ruído em séries temporais.

Na análise de Fourier, temos perda da informação no tempo quando é feita a transformação para o domínio da frequência. A falta de localização do sinal no tempo permite apenas uma análise do comportamento global dos sinais. Para sinais estacionários, esta característica não é muito importante. No entanto, quando o sinal contém características não estacionárias ou transitórias, tais como tendências e mudanças abruptas, a transformada de Fourier apresenta uma séria desvantagem.

Uma forma de olhar para a diferença da resolução tempo-frequência entre as duas técnicas é através da cobertura das funções base no plano tempo-frequência [25]. Na transformada de Fourier, um quadrado com uma certa largura truncando as funções seno e cosseno forma uma janela que é igual em tamanho em todas frequências. Isto faz com que a resolução da análise seja a mesma em todos os locais do plano tempo-frequência.

Uma vantagem da transformada *wavelet* é que essa janela não é constante. Para isolar descontinuidades, seria útil ter funções base de tamanho pequeno, enquanto para obter análises de frequência detalhada, funções base longas são desejáveis. Uma maneira de obter isto é ter funções base de alta-frequência pequenas e outras de baixa-frequência que sejam longas [16].

Outra vantagem na análise *wavelet* é que os coeficientes têm um parâmetro local. Isto é necessário para distinguir coeficientes diferentes no mesmo nível, o que contrasta com a transformada de Fourier onde a informação acerca das frequências mais relevantes na série é obtida, mas é difícil distinguir onde estão presentes.

De notar, como dito anteriormente, que a WFT também consegue detetar propriedades locais, pois a transformada de Fourier discreta é aplicada a pequenas partes da série. Akin (2002) comparou os dois métodos aplicando-os na análise de sinais de eletrocardiograma cerebral e sugere que o método *wavelet* ofereceu melhores resultados [18].

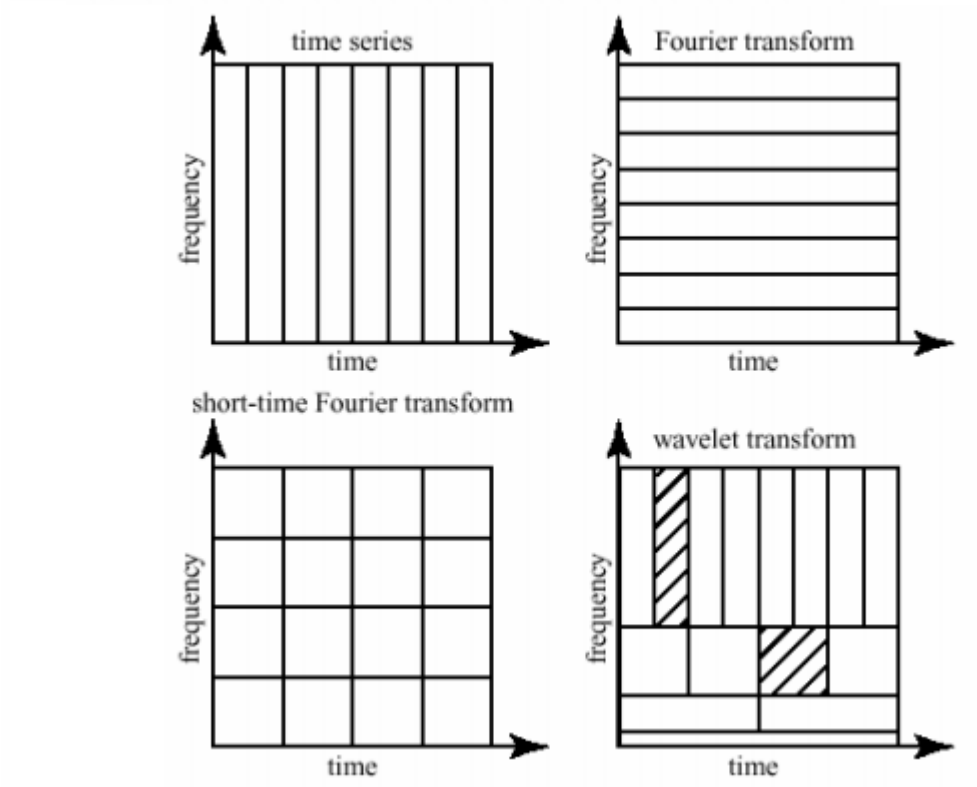


Figura 3.6: Comparação da resolução tempo-frequência entre as diferentes transformadas. Na transformada de Fourier de curto termo nota-se um tamanho da janela independente da resolução tempo-frequência, ao contrário da transformada *wavelet* onde o tamanho dessa janela se altera de acordo com a resolução [2].

3.2.4 Transformada *Wavelet* Discreta

A transformada *wavelet* contínua (CWT) é útil, por exemplo, na construção de um gráfico de três eixos (tempo, escala/nível e uma outra, normalmente representado por cores ou diferentes níveis de brilho) chamado de escalograma, pois a janela de análise é dimensionada e posicionada em qualquer posição. Esta flexibilidade permite a criação de uma imagem mais suave e agradável na visualização tanto em tempo como em escala (análogo à frequência). A CWT é uma transformada redundante porque a janela de análise pode sobrepor-se. De facto, CWT é considerada infinitamente redundante. Por outro lado, DWT é uma transformada não redundante. Foi desenvolvida para que houvesse uma correspondência de um para um entre a informação no domínio do sinal e no domínio da transformada. Esta correspondência é útil na reconstrução do sinal, mas as janelas de análise são fixas tanto no tempo como em escala, fazendo com que o escalograma não seja tão satisfatório para análise visual do sinal [2].

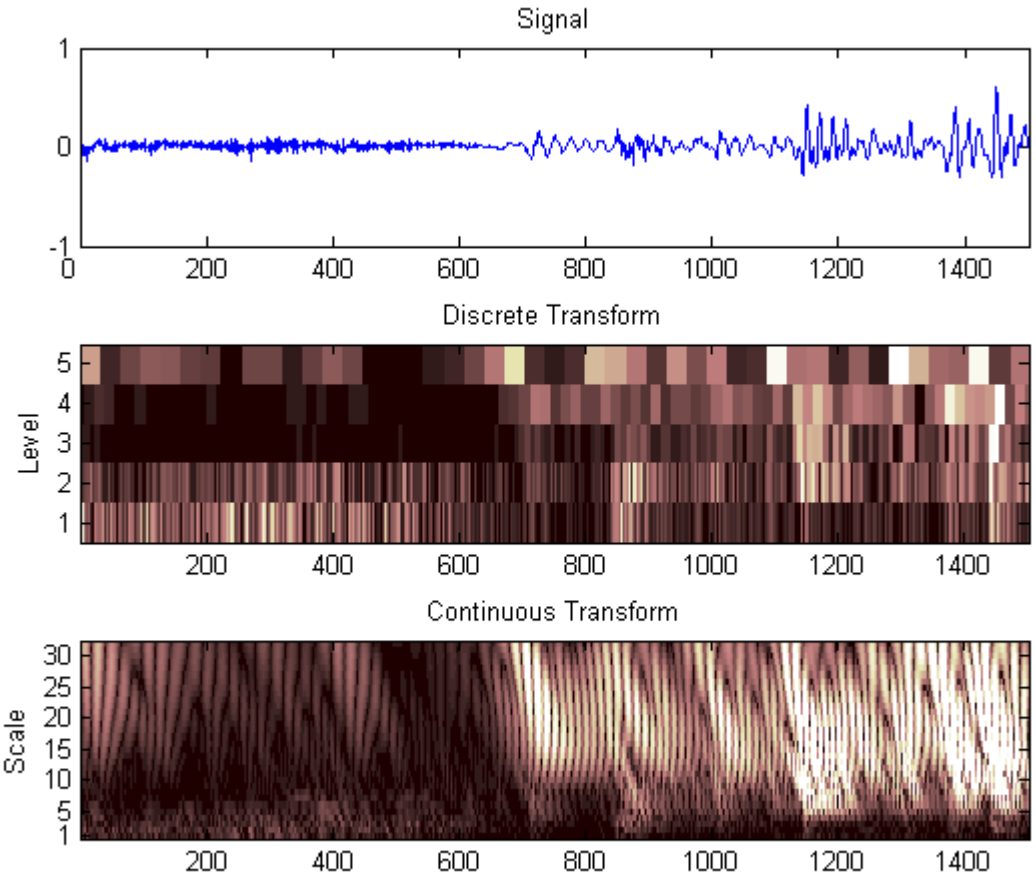


Figura 3.7: Escalograma: Comparação entre CWT e DWT.

Por outro lado, DWT tem enormíssimas vantagens, em comparação com CWT: é capaz de reduzir significativamente o tempo de computação; é de fácil implementação e fornece informação suficiente tanto para análise como para reconstrução do sinal; analisa o sinal decompondo-o em informação aproximada e detalhada [2].

Temos então um sinal que é transformado, através de DWT (e da escolha de uma função base *wavelet*), de um domínio temporal para um domínio de tempo-frequência. É utilizada uma decomposição em múltiplos níveis onde o sinal é inicialmente decomposto em aproximação e detalhe conforme a árvore representada a seguir.

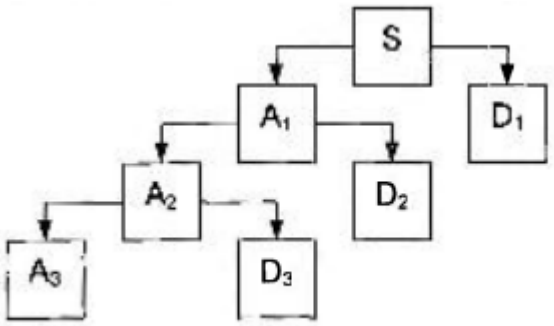


Figura 3.8: Árvore de decomposição *wavelet*. Em D_1 estarão representadas as frequências mais altas. Para $i > 1$, em D_i teremos as representações das frequências cada vez mais pequenas [2].

Os valores do output são chamados de coeficientes. Os coeficientes de baixo valor absoluto são considerados ruído, e os de alto valor absoluto têm associada muita informação comparativamente à quantidade de ruído. Num segundo passo, são anulados todos os coeficientes abaixo de um certo valor pré-definido, chamado de *threshold*, enquanto que os restantes coeficientes permanecerão inalterados (*hard threshold*) ou sofrerão um *soft threshold*. O último passo é a aplicação da transformada inversa (IDWT), que vai reconstruir o sinal [26].

A expressão (3.7) tem uma variante discreta. Por uma questão de facilidade na explicação do método, vamos reescrever a equação da mãe-*wavelet*, fazendo uma mudança de variável da seguinte forma:

$$\frac{1}{\sqrt{j}} = 2^{\frac{j^*}{2}} \tag{3.20}$$

onde j* é a nova variável de escala/resolução. Continuemos, porém, a chamar-lhe j, e a equação da mãe *wavelet* fica

$$\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k) \tag{3.21}$$

Um ponto de começo na tentativa de pereber as *wavelets* é através das suas funções base. Tal como Fourier fazia uso das funções seno e cosseno, uma transformada *wavelet* tem uma função mãe, $\psi(x)$, e uma função pai, $\phi(x)$ (mencionadas na equação (3.7)). Estas funções estão relacionadas através das seguintes equações:

$$\psi(x) = \sum_{k \in Z} g_k \sqrt{2}\phi(2x - k). \tag{3.22}$$

$$\phi(x) = \sum_{k \in Z} h_k \sqrt{2} \phi(2x - k). \tag{3.23}$$

O conjunto de coeficientes $\{g_k\}$ contém os filtros passa-altas (tendem a manter as altas frequências e a eliminar frequências baixas) associados a uma função *wavelet* definida. Usando ambas as funções, é possível representar uma série discreta representada em (3.7), reescrita da seguinte forma:

$$f(x) = c_{0,0}\phi_{0,0}(x) + \sum_{k} \sum_{j>0} d_{j,k}\psi_{j,k}(x) \tag{3.24}$$

Tal como já foi referido, os índices k e j estão associados à translação e escala, respetivamente. Referem-se ao posicionamento e nível de detalhe dos coeficientes ao longo do sinal.

O conceito de nível de detalhe é muito importante nas *wavelets*. Para a DWT, o comprimento da série a analisar tem de ser uma potência de dois. Numa série de comprimento 2^J existem J diferentes níveis de detalhe, variando entre 0 e $J-1$, onde o nível J contém a série original e a escala $J-1$ os melhores coeficientes de escala. Quando $j = 0$ temos a função base o mais esticada possível, cobrindo a série na sua totalidade. Consequentemente, nesse nível teremos apenas um coeficiente. É claro que, representar toda a série com um só coeficiente é uma aproximação muito fraca. Se, em vez disso, $j = 1$, a função *wavelet* vai cobrir apenas metade da série e isto vai resultar em dois coeficientes. Naturalmente, os detalhes da série estarão mais bem representados do que quando $j = 0$. Este processo continua, duplicando o número de coeficientes à medida que o valor de j aumenta, ate que $j = J$.

O próximo passo é perceber como é que estes coeficientes são calculados. Este processo é conhecido como algoritmo de pirâmide. Primeiro são calculados os coeficientes referentes ao pai-*wavelet* e, posteriormente, os referentes à mãe-*wavelet*. Os primeiros, $\{c_{j,k}\}$, são conhecidos como os coeficientes *smooth* e são usados para representar o nível da série. São gerados através de um processo similiar ao da *weighted moving average*, da seguinte forma:

$$c_{j-1,k} = \sum_{n} h_{n-2k} c_{j,n}. \tag{3.25}$$

Os coeficientes *wavelet* $\{h_k\}$, que são análogos aos pesos da *moving average*, são filtros passa-baixas (tendem a eliminar altas frequências e a manter frequências baixas).

Estes são específicos a cada base *wavelet* usada na transformada. O número de coeficientes também varia consoante a base usada. No entanto, todas têm de obedecer à propriedade $\sum_k h_k^2 = 1$. Como se vê a partir da equação (3.25), os coeficientes de uma certa escala são calculados a partir dos coeficientes de escala superior.

Os coeficientes associados à mãe-*wavelet* são chamados coeficientes de detalhe. São usados para representar a diferença entre a série e os coeficiente $c_{j,k}$. São calculados da seguinte forma:

$$d_{j-1,k} = \sum_n g_{n-2k} c_{j,n}. \tag{3.26}$$

Embora pareçam muito similares aos coeficientes associados ao pai-*wavelet*, existem algumas diferenças importantes. Primeiro, os coeficientes h_k, que são filtros passa-baixas, foram substituídos pelos filtros passa-altas $g_{j,k}$. Estes sãos os mesmo coeficientes usados na equação (3.22). Estes dois conjuntos de coeficientes estão relacionados da seguinte forma:

$$g_k = (-1)^k h_{1-k}. \tag{3.27}$$

A introdução de sinais negativos no filtro dá aos coeficientes de detalhe capacidade de mostrar mais detalhe em como o processo está a mudar e em como a série difere da representação dos coeficientes *smooth*. Outra característica a realçar é que tanto na equação (3.25) como na equação (3.26), apenas os coeficientes *smooth* são usados. Isto significa que apenas estes são necessários na derivação dos coeficientes *smooth* e detalhe para o próximo nível de decomposição.

Este procedimento está sumariado na figura seguinte. À medida que a transformada é executada, os coeficientes de detalhe são guardados, ao contrário dos coeficientes *smooth* (exceto no nível mais baixo). Isto acontece porque a série pode ser representada usando apenas estes coeficientes, como representado na equação (3.24).

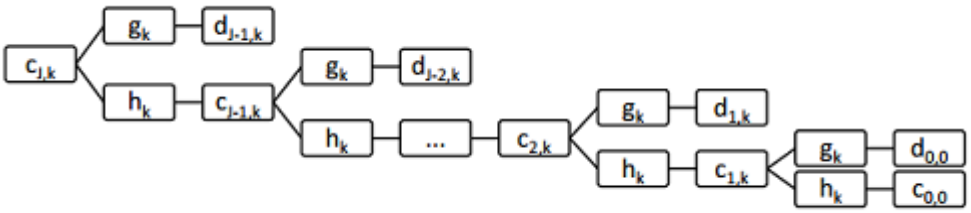


Figura 3.9: Estrutura de computação dos coeficientes wavelet

A computação não tem obrigatoriamente de ser executada para todo o nível j. No entanto, o coeficiente *smooth* do último nível calculado terá de ser guardado. À primeira vista, parece perdermos informação importante por apenas guardarmos os coeficientes de detalhe (e um *smooth*), mas tal não é verdade. Os coeficientes *smooth* apenas descrevem o nível da série. Durante a análise, esta característica não é muito relevante comparada com características como quebras, saltos, etc. Estes detalhes estão presentes nos coeficientes de detalhe [18]. Este algoritmo eficiente requer apenas $O(n)$ operações (ver Mallat [27]).

Uma abordagem alternativa na construção dos coeficientes é atraves de matrizes. Como o algoritmo piramidal consiste basicamente em séries de multiplicações, adições e subtrações, não será muito difícil de imaginar o mesmo ser conseguido através da multiplicação de matrizes.

Dado um conjunto de dados $f_1, ..., f_n$, com $n = 2^J$, que formam um vetor coluna f, existe uma matriz ortogonal W $(n \times n)$ tal que a transformada discreta *wavelet* θ_{jk} é dada por

$$\theta = Wf, \tag{3.28}$$

onde θ é o n-vetor dos coeficientes da *wavelet* discreta $\theta_{jk}, j = 0, ..., J-1$ e $k = 1, ..., 2^j$. A transformada inversa pode ser representada da seguinte forma:

$$f = W^T\theta \tag{3.29}$$

Usando a fórmula da transformada da inversa, é possível notar que as linhas de W correspondem à discretização da mãe-*wavelet* em várias escalas e translações diferentes. Donoho e Johnstone [28] notaram a seguinte aproximação:

$$\sqrt{n}W_{j,k}(i) \approx 2^{\frac{j}{2}}\psi(2^j t - k),\ t = \frac{i}{n}, \tag{3.30}$$

onde $W_{j,k}(i)$ é o i-ésimo elemento da linha (j,k) de W. Pode ser visto que o coeficiente θ_{jk} quantifica a contribuição das funções de base $W_{j,k}$ que estão localizadas num intervalo de tamanho 2^{-j} e frequência 2^j. Dito de outro modo, θ_{jk} indica a quantidade de sinal à volta de 2^{-j} e perto da frequência 2^j [17]. O vetor θ contém todos os coeficientes de detalhe de todos os níveis e o coeficiente de aproximação do nível mais baixo, tendo então comprimento n e a informação necessária para reconstruir a série. Como este método envolve a multiplicação de uma matriz, vai requerer O(n^2) operações, sendo assim um processo mais lento do que aquele visto com o algoritmo piramidal que devolve exatamente a mesma informação. A razão para isto acontecer é que muitos dos coeficientes da matriz têm de facto valor nulo, o que implica que as suas operações não sejam necessárias [18].

3.2.5 Funções Base

Um fator importante a ter em conta quando se fala em análise por *wavelets* é o da escolha da função base. Não existindo uma escolha perfeita para todos os casos, as várias áreas de aplicação têm feito escolhas diferentes [29]. Para uma função poder ser usada na análise *wavelet*, tem de satisfazer certas propriedades. Iremos descrever algumas delas (ver VidaKovic (1999) para mais detalhes).

A análise *wavelet* tem de permitir uma decomposição em várias escalas. Isto só é possível se a função pai-*wavelet* puder ser construída no nível 0 através duma combinação linear de funções de nível 1, isto é,

$$\phi_{0,0} = \sum_k h_k \phi_{1,k}(x), \tag{3.31}$$

onde $\{h_k\}$ são os filtros já mencionados atrás.

É também necessário que a base seja normalizada e ortogonal. No primeiro temos de garantir que

$$\sum_k h_k = \sqrt{2}, \tag{3.32}$$

enquanto para a ortogonalidade temos

$$< \phi_{j,k}(x), \phi_{i,l}(x) > = \int \phi_{j,k}(x)\phi_{i,l}(x)\, dx = \delta_{ij}\delta_{lk}. \tag{3.33}$$

Uma consequência disto é que os coeficientes de filtro também têm de ser ortogonais, satisfazendo

$$\sum_k h_k h_{k-2l} = \delta_l, \tag{3.34}$$

e quando temos $l = 0$, fica

$$\sum_k h_k^2 = 1. \tag{3.35}$$

Se optarmos pela perspetiva matricial, o que esta propriedade garante é que

$$W^T W = W W^T = I \tag{3.36}$$

A base *wavelet* mais simples que existe é a Haar (Haar, 1910). Esta *wavelet* foi a primeira a ser usada, e tem como função *wavelet* e função de escala as seguintes definições, respetivamente:

$$\psi(x) = \begin{cases} -1 & x \in [0, \frac{1}{2}[\\ 1 & x \in [\frac{1}{2}, 1[\\ 0 & \text{caso contrário} \end{cases}$$

$$\phi(x) = \begin{cases} 1 & x \in [0, 1] \\ 0 & \text{caso contrário} \end{cases}$$

A simplicidade deste *wavelet* significa que é ideal na demonstração de princípios básicos no que toca ao processo de transformada. A equação de escala (3.23), é muito simples para esta *wavelet* e os filtros associados a esta base são construídos da seguinte forma

$$\begin{aligned} \phi(x) &= \phi(2x) + \phi(2x-1) \\ &= \frac{1}{\sqrt{2}}\sqrt{2}\phi(x) + \frac{1}{\sqrt{2}}\sqrt{2}\phi(2x-1), \end{aligned} \tag{3.37}$$

onde se deduzem

$$h = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \quad g = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \tag{3.38}$$

Quando estes filtros são aplicados nas equações (3.25) e (3.26), os cálculos são simplesmente uma soma ou subtração dividida pela raiz de dois.

Como demonstração, vamos aplicar o DWT a uma série de comprimento 8 (isto significa que temos $J = 3$ níveis de coeficientes. A primeira linha representa a série original e as linhas seguintes são os respetivos coeficientes aplicando a *wavelet* Haar. Enquanto os coeficientes *smooth* dum determinado nível são simplesmente a soma dos pares correspondentes do nível superior, os de detalhe são a diferença entre eles. Notamos que cada linha foi multiplicada por $\sqrt{2}$ para facilitar a visualização dos cálculos.

$$
\begin{aligned}
c_{3,k} &= 3 \quad 2 \quad 5 \quad 1 \quad 3 \quad 6 \quad 2 \quad 8 \\
\sqrt{2}c_{2,k} &= \quad 5 \quad\quad 6 \quad\quad 9 \quad\quad 10 \\
\sqrt{2}d_{2,k} &= \quad 1 \quad\quad 4 \quad\quad -3 \quad\quad -6 \\
2c_{1,k} &= \quad\quad 11 \quad\quad\quad\quad 19 \\
2d_{1,k} &= \quad\quad -1 \quad\quad\quad\quad -1 \\
2\sqrt{2}c_{0,k} &= \quad\quad\quad\quad 30 \\
2\sqrt{2}d_{0,k} &= \quad\quad\quad\quad -8
\end{aligned}
$$

A aproximação à série original, usando a equação (3.24), fica

$$
\begin{aligned}
f = \quad & \frac{15}{\sqrt{2}}\phi_{0,0} - \frac{4}{\sqrt{2}}\psi_{0,0} - \frac{1}{2}\psi_{1,0} - \frac{1}{2}\psi_{1,1} + \\
& \frac{1}{\sqrt{2}}\psi_{2,0} + \frac{4}{\sqrt{2}}\psi_{2,1} - \frac{3}{\sqrt{2}}\psi_{2,2} - \frac{6}{\sqrt{2}}\psi_{2,3}
\end{aligned}
\tag{3.39}
$$

Embora esta *wavelet* proporcione exemplos fáceis de entender, e é a única *wavelet* capaz de criar funções perfeitamente simétricas ou assimétricas que tenham um suporte compacto, tem a limitação de ser pouco regular [18, 29]

Apenas nos anos oitenta, novas famílias de *wavelets* foram desenvolvidas, aumentando assim o número de funções base disponível.

O grupo de *wavelets* desenvolvido por Daubechies (1998) é possivelmente o mais usado hoje em dia. Cada *wavelet* tem um certo número de momentos de fuga, que pode variar entre 1 e 10, e para cada estão associados dois coeficientes de filtro. Podemos representar cada *wavelet* como Dj (j é o número de coeficientes) ou dbi (i é o número de momentos). O momento de fuga refere-se à capacidade da *wavelet* de representar comportamento polinomial ou informação no sinal. Por exemplo, db1 (que é a *wavelet* Haar), facilmente representa funções constantes; db2 facilmente representa constantes e funções lineares, etc. Formalmente, é dito que tem m momentos de fuga se satisfizer

$$
\int x^l \psi(x)\, dx = 0, \quad \text{para } l = 0, ..., m-1. \tag{3.40}
$$

Se uma *wavelet* com m momentos de fuga for usada para analisar uma função polinomial de ordem m, então todos os coeficientes de detalhe serão zero. Isto é útil se a

série tem uma estrutura polinomial suave com uma quebra ocasional. Com a escolha de uma *wavelet* com o número apropriado de momentos de fuga, a maior parte dos coeficientes será zero, com exceção daqueles que representam as quebras [18]. Estamos perante uma *wavelet* com boa ortogonalidade e regularidade, ao contrário da simetria [29].

A *wavelet* Coiflet é uma alternativa que garante não só o desaparecimento da alta distância como também tem boa simetria, isto sob o sacrifício do comprimento do conjunto alcançado [29].

Outra *wavelet* usada é a Morlet, que tem como grande vantagem o tamanho da janela. Nem todas as funções *wavelets* possuem uma expressão explícita, mas a par da função Haar, a Morlet possui uma:

$$\Psi(x) = c_\omega \pi^{-\frac{1}{4}} e^{-\frac{t^2}{2\sigma^2}} (e^{i\omega t} - e^{-\frac{1}{2}\omega^2},$$
$$c_\omega = (1 - e^{-\omega^2} - 2e^{-\frac{3}{4}\omega^2})^{-\frac{1}{2}} \tag{3.41}$$

onde $\omega > 0$ é a frequência base e $\sigma > 0$ (ver Daubechies, 1992 para mais detalhes).

Na figura seguinte, vemos a influência da escala e do carácter de uma oscilação local na *wavelet* Morlet, com a aplicação de três escalas diferentes e translação nula [19].

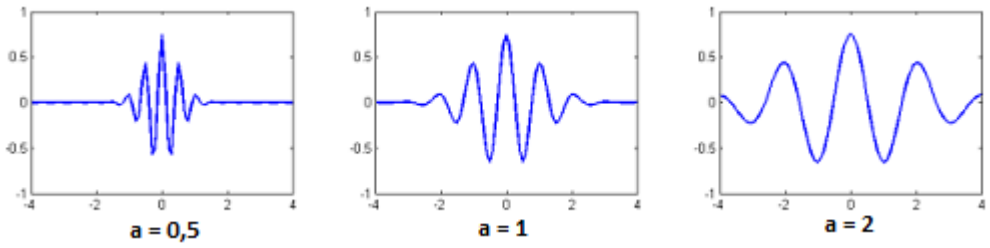


Figura 3.10: Wavelet Morlet em escalas diferentes.

3.2.6 *Wavelet* Denoising

Em investigação científica, a relação entre teoria e experimentação deveria fornecer um progresso conjunto significativo, mas muitas vezes, existe uma fratura entre teóricos e experimentalistas. Nomeadamente em estatística são provados teoremas com base em princípios assimptóticos que irrealisticamente requerem uma quantidade infinita de dados, enquanto experimentalistas realizam experiências com base ou em dados

reais ou sintetizados, onde o número de dados é finito. A questão que se coloca é quando é que os teoremas de grande escala se aplicam às experiências de escala curta. A resposta a esta questão encontra-se no teorema do limite central, isto é, à medida que o tamanho amostral aumenta, a distribuição amostral da sua média aproxima-se cada vez mais da distribuição normal.

No que toca a *wavelet shrinkage denoising*, as justificações teóricas e argumentos a seu favor têm sido muito atraentes. O procedimento não requer qualquer suposição acerca da natureza do sinal, permite descontinuidades e variação espacial no sinal, e explora propriedades da multiresolução essenciais na transformada *wavelet*. Mais ainda, o procedimento mapeia ruído branco do sinal em ruído branco no domínio *wavelet*. Assim, enquanto a energia do sinal se concentra em menos coeficientes no domínio *wavelet*, o ruído não. Este princípio permite a separação do sinal do ruído [11].

A determinação do procedimento mais adequado envolve necessariamente experiências no sentido de comparar a performance do método em questão (tendo em conta a combinação entre os parâmetros da transformada *wavelet*, regras de remoção de ruído e threshold para o intervalo de dados e ruído esperados). É pouco provável que um procedimento de *wavelet shrinkage denoising* em particular seja apropriado ou ótimo para todos os problemas práticos. No entanto, é muito provável que muitos problemas sejam resolvidos eficientemente, após experimentações adequadas, usando métodos *wavelet* de remoção de ruído [11].

A técnica não-linear introduzida por David Donoho, *wavelet shrinkage denoising*, funciona do seguinte modo: quando decompomos os dados de entrada aplicando *wavelets*, usamos filtros já referidos anteriormente. Alguns desses resultam em coeficientes que correspondem ao detalhe desse conjunto de dados. Se esses coeficientes forem pequenos podem ser omitidos sem que as propriedades gerais do sinal sejam substancialmente afetadas. A ideia de *thresholding* é anular todos esses coeficientes que são menores que um valor pré-definido. Posteriormente, através da transformada inversa, obtemos o sinal reconstruido. O resultado é um sinal limpo cujos detalhes importantes ainda são mostrados [16].

Formalmente vamos definir o processo do seguinte modo, com input no tempo t,

$$X(t) = S(t) + N(t), \tag{3.42}$$

onde $S(t)$ é o sinal depois de removido o ruído $N(t)$. Denotem-se $W(.)$ e $W^{-1}(.)$ a transformada *wavelet* e transformada *wavelet* inversa, respetivamente e seja $D(., \lambda)$ o

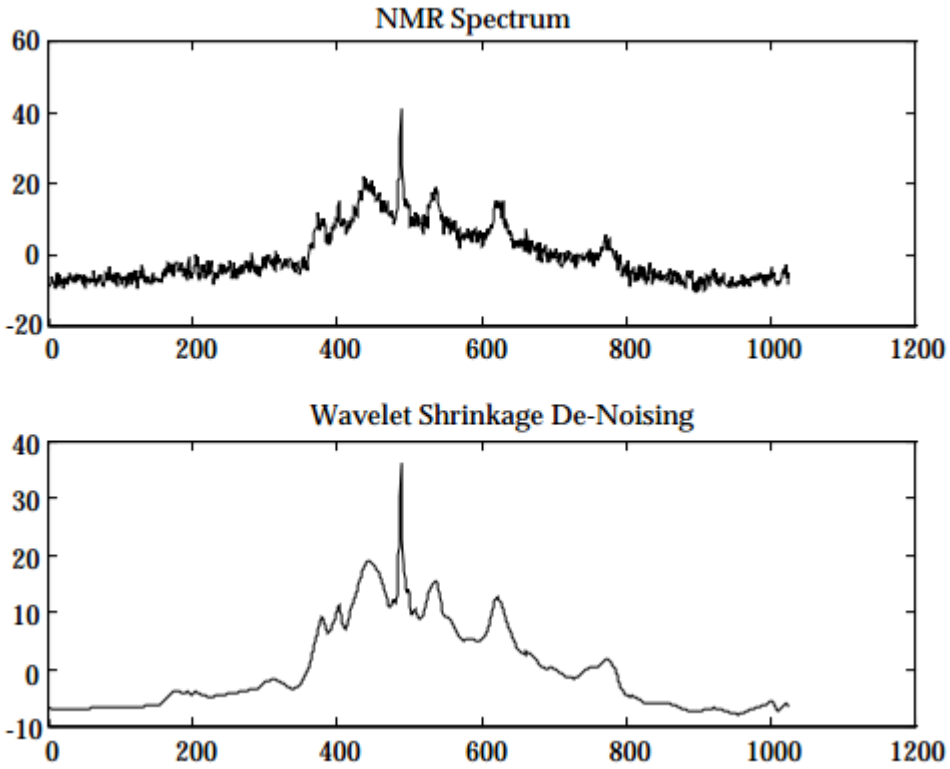


Figura 3.11: Sinal de ressonância magnética nuclear (NMR). Comparação entre o sinal original (topo) e o sinal após a remoção do ruído (em baixo). (Imagem de David Donoho, Standford University, dados NMR são da autoria de Adrian Maudsley, VA Medical Center, San Francisco).

operador de remoção de ruído, com *threshold* λ. No sentido de obter a aproximação $S'(t)$ do sinal $S(t)$ são efetuados os seguintes passos:

$$Y = W(X) \tag{3.43}$$

$$Z = D(Y, \lambda) \tag{3.44}$$

$$S' = W^{-1}(Z) \tag{3.45}$$

Enquanto o primeiro e terceiro passos já foram referidos em capítulos anteriores, vamos abordar o que acontece na equação (3.44).

Desde o trabalho de Donoho e Johnstone (1994, 1995), tem havido muita investigação na tentativa de encontrar o melhor valor de *threshold*. Nason (1996) obteve um valor de *threshold* através de *cross validation* e o procedimento de hipótese múltipla é sugerido por Abramovich e Benjamin (1995). Um valor pequeno de *threshold* vai resultar num *output* mais próximo do sinal original mas será, ainda assim, um resultado com

ruído. Por outro lado, se for demasiado elevado, embora vá produzir um sinal sem ruído, certamente irá eliminar informação relevante do sinal [26]. Aqui surgem duas questões: Como definir o *threshold* ideal? Após seleção do *threshold*, como aplicá-lo nos coeficientes? Comecemos pela segunda. Vamos aqui abordar dois métodos para lidarmos com os coeficientes, através de *hard thresholding* ou *soft thresholding*.

Dado coeficiente s', podemos definir *hard thresholding* e *soft thresholding*, respetivamente, como

$$T_{\text{hard}}(s';\lambda) = s' \cdot I(|s'| > \lambda), \tag{3.46}$$

$$T_{\text{soft}}(s';\lambda) = sgn(s') \cdot (|s'| - \lambda) \cdot I(|s'| > \lambda), \tag{3.47}$$

onde $sgn(s')$ representa o sinal de s' e I a função indicadora.

Não havendo grande teoria desenvolvida, favorecendo uma ou outra técnica, decidimos aplicar *hard thresholding* nos nossos testes.

Voltando à primeira questão, como podemos encontrar o *threshold* ideal? Como já foi mencionado, esta escolha é muito importante no sentido de evitar valores demasiado baixos/altos que não irão fornecer os resultados esperados. Vamos descrever alguns dos métodos mais conhecidos:

Universal threshold

Donoho e Johnstone [28] propuseram este método universal, que utilizado com *hard thresholding* é também conhecido por *VisuShrink*, e é dado por

$$T_{UV} = \sigma' \sqrt{2 \log |S'|}, \tag{3.48}$$

onde $\log |S'|$ é logaritmo natural do tamanho do conjunto de dados e σ' uma aproximação do nível de ruído σ, que normalmente é assumido seguir uma distribuição normal e é calculado através da mediana do desvio absoluto mediano dos coeficientes *wavelet* (MAD), isto é,

$$\text{MAD}_{S'} = \text{mediana}(|s'_i - \text{mediana}(s')|). \tag{3.49}$$

Uma propriedade do *VisuShrink* é que quase garante uma reconstrução livre de ruído, pois anula bastantes coeficientes. Isto tem o custo de uma reconstrução menos precisa [17].

SURE threshold

Baseado no estimador de risco de Stein, Donoho e Johnstone (1994) propuseram outra escolha para o valor do *threshold*. Este método tem a particularidade de escolher um *threshold* para cada nível de resolução na trasnformada *wavelet* [17]. Como este processo não obteve bons resultados em matrizes esparsas, propuseram uma alternativa à qual chamaram *SureShrink*. Aqui, era utilizado o *threshold* universal referido anteriormente se

$$\sum_k (WT(j,k)^2 - 1) \leq \log_2 |S'|^{\frac{3}{2}}, \tag{3.50}$$

onde WT é a transformada *wavelet*, e o *threshold* SURE caso contrário. Uma diferença em relação ao *VisuShrink*, é que este método utiliza *soft thresholding* [19].

Métodos de Ogden

Entre outras ideias, Ogden desenvolveu dois métodos como opção de escolha para o valor do *threshold*, dadas pelo nome de *selection thresholding* e *data-analytic thresholding*. A primeira é baseada em testes de hipótese dos coeficientes, nível por nível. A segunda é também abordada tendo em conta a escala, mas a diferença é que aqui são analisadas graficamente as somas cumulativas do quadrado dos coeficientes, para cada nível. Através de um processo *Brownian bridge*, os coeficientes com valor mais elevado são retirados do teste, até que deixe de haver coeficientes que se sobressaiam em relação aos outros. Estes são considerados ruído. Uma vantagem do método *data-analytic thresholding* é a de não separar coeficientes adjacentes no tempo. Por exemplo, descontinuidades podem fazer com que dois coeficientes adjacentes tenham valor alto (em vez de um), e este método identifica-os como um.

Existem outros métodos, também muito usados, como o *minimax* (ver [28]), métodos Baysianos ou através de *cross validation* [17], etc.

Em virtude de não haver muita pesquisa no que toca ao uso de *wavelets* em traçados fetais e por uma questão de facilidade de programação foi usado o métdodo *VisuShrink* em todos os testes efetuados, isto é, a combinação entre *universal threshold* e *hard threshold*.

Capítulo 4

Análise de Resultados

Nos testes realizados foram usados *datasets* diferentes, embora todos tenham em comum o facto de conterem ficheiros pré-processados pelo sistema Sisporto, contendo não só a frequência cardíaca do feto, mas também o tempo de intervalo entre cada registo, entre outras características menos relevantes para o nosso caso. Posteriormente, fizemos uma filtragem do conteúdo desses ficheiros de forma a que os ficheiros contenham apenas uma coluna, a da frequência cardíaca. Todos estes registos ditam uma diferença de 250 milissegundos da anterior.

Por questões técnicas na realização dos cardiotocogramas, por vezes acontece obter valores de frequência cardíaca iguais ou próximos de zero. No sentido de evitar distorções na análise dos resultados, estes valores foram apagados dos ficheiros.

Definimos rácio de compressão a divisão entre o tamanho da versão original do ficheiro e o tamanho da sua versão comprimida. A taxa de compressão é o seu inverso e pode ser vista como a percentagem do traçado comprimido em relação ao original.

Os gráficos foram criados com recurso ao OpenOffice, as tabelas e os *boxplots* usando o SPSS e as árvores de classificação usando o *CompLearn*.

4.1 Bzip2 vs Gzip vs Paq8l

O conjunto de dados usado para este teste é formado por 1406 traçados de batimento cardíaco fetal de 250 grávidas. Os cardiotocogramas foram realizados entre as semanas 24 e 40 da gravidez e pretendemos comparar o comportamento de diferentes compres-

sores, assim como analisar certas características do feto tais como o género, peso, se o nascimento foi prematuro e se o estado de saúde à nascença é considerado normal ou não. Os resultados foram divididos por semana e os gráficos contêm as médias da taxa de compressão para cada uma dessas semanas.

Para este conjunto de dados, além do pré-processamento já referido, e em virtude de nem todos os traçados terem o mesmo comprimento, decidimos uniformizar de forma a que todos tivessem apenas 10 minutos de traçado, o que equivale a 2400 entradas por ficheiro.

Numa primeira fase, foram selecionados os traçados referentes aos fetos considerados normais após o parto, através do índice Apgar. Tal como previsto, a compressão por parte do compressor paq8l é maior comparativamente ao gzip e bzip2, obtendo assim taxa de compressão menor (Figura 4.1).

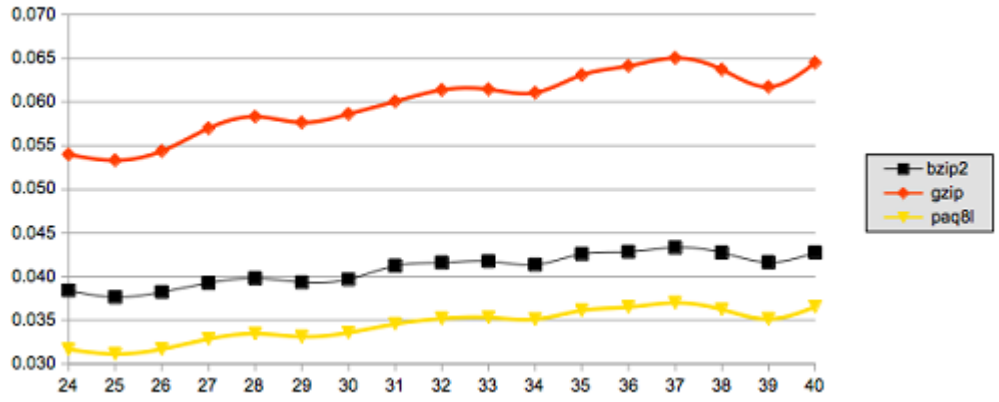


Figura 4.1: Médias da taxa de compressão, por semana, usando os três compressores.

Por outro lado, como seria de esperar, o tempo e memória necessários para a compressão são muito mais elevados no caso do compressor paq8l.

De seguida foi dividido o conjunto de dados em dois grupos: Os traçados de bebés considerados saudáveis à nascença, e aqueles considerados patológicos (com base no pH do cordão umbilical à nascença). Pela análise da Figura 4.2 notamos não só que os compressores comportam-se de forma quase idêntica, mas também uma maior capacidade de compressão, salvo exceções, nos traçados de fetos patológicos, embora a dinâmica ao longo das semanas entre os dois grupos seja similar.

Figura 4.2: Gráfico das médias da taxa de compressão semanais dos traçados dos fetos considerados normais e patológicos usando os três compressores.

Com o mesmo conjunto de dados, dividindo o grupo por género, notámos que a menos da diferença da taxa de compressão já mencionada, os três compressores comportam-se de forma similar para cada um dos géneros. De salientar uma diferença substancial entre os géneros na semana 27 (Figura 4.3).

Figura 4.3: Gráfico das médias da taxa de compressão semanais, por género, dos traçados dos fetos considerados normais usando os três compressores.

O mesmo foi feito para os fetos considerados patológicos e embora o comportamento dos compressores seja similar, notámos uma diferença no comportamento a partir da semana 36 entre os géneros (Figura 4.4).

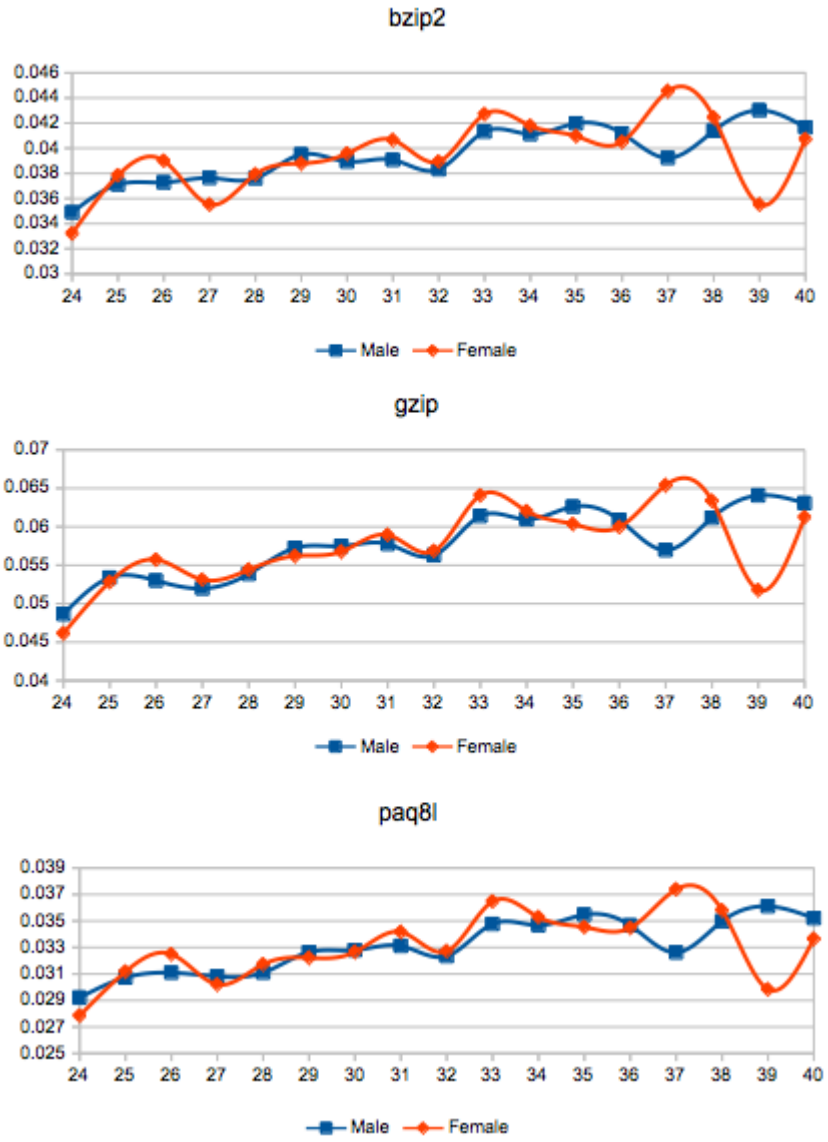


Figura 4.4: Gráfico das médias da taxa de compressão semanais, por género, dos traçados dos fetos considerados patológicos usando os três compressores.

No que toca à prematuridade do parto, dividimos o *dataset* em traçados cujo parto foi prematuro ou não (definimos como prematuro o bebé que tivesse nascido até às 36 semanas). Verificámos que em todos os compressores, a taxa de compressão era maior nos prematuros, notando-se maiores oscilações nos pré-termo (Figura 4.5).

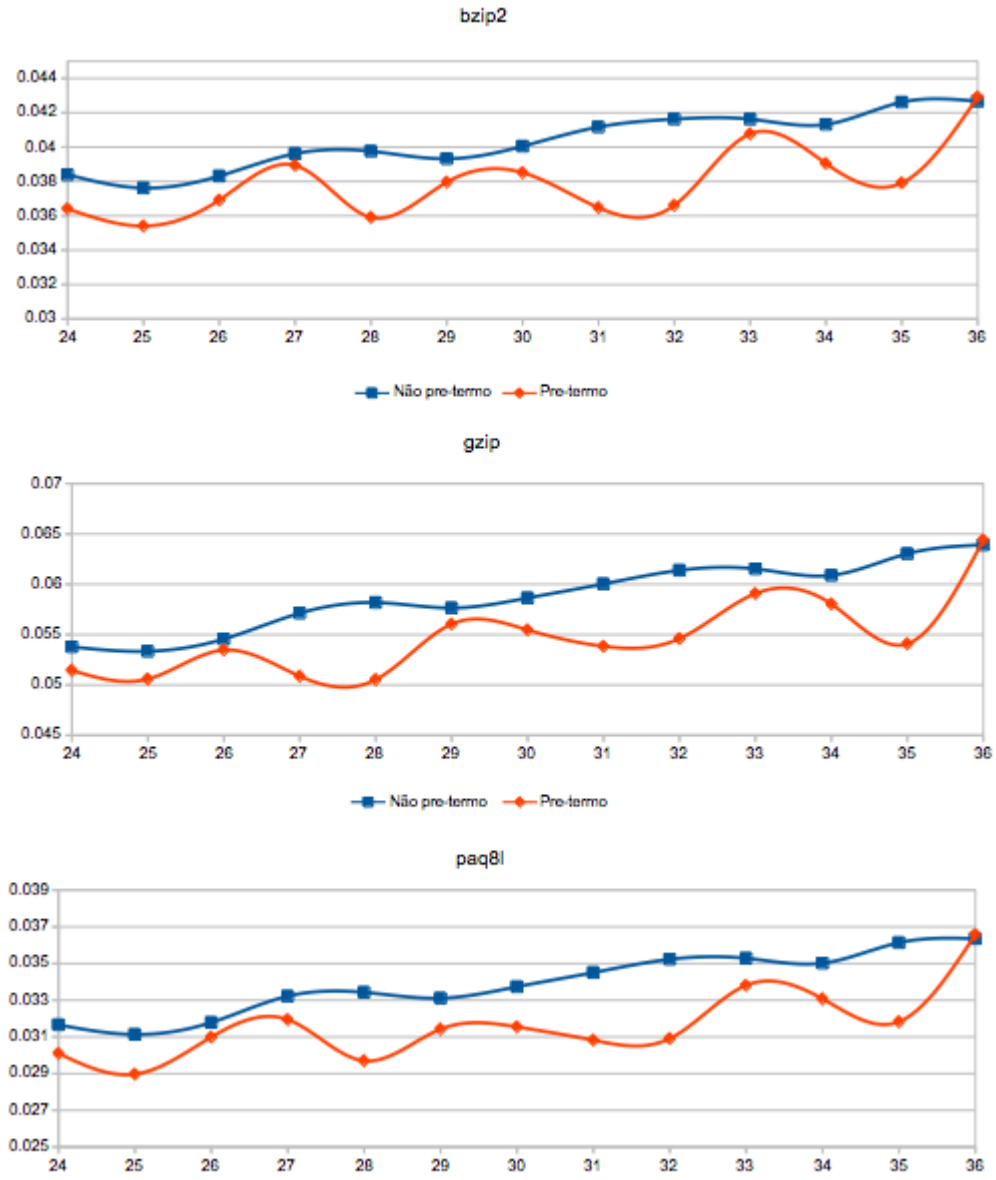


Figura 4.5: Gráfico das médias da taxa de compressão semanais, dos traçados dos fetos considerados prematuros e não prematuros usando os três compressores.

A última característica analisada foi o peso à nascença. Esta variável foi analisada em quatro diferentes variantes: os bebés cujo peso à nascença é considerado normal, aqueles com peso abaixo de um certo percentil, isto é, percentil 3 (p3) e percentil 10 (p10) e os que se encontram entre os dois.

Mais uma vez, a menos da diferença do poder de compressão, os 3 compressores comportam-se de forma similar. Notámos uma certa estabilidade na evolução dos

traçados considerados normais, ao contrário dos restantes. De salientar as grandes diferenças registadas na semana 30 onde enquanto a taxa dos traçados normais e p3 são similares, os traçados p10 e "entre p3 e p10"são muito mais baixos (Figura 4.6).

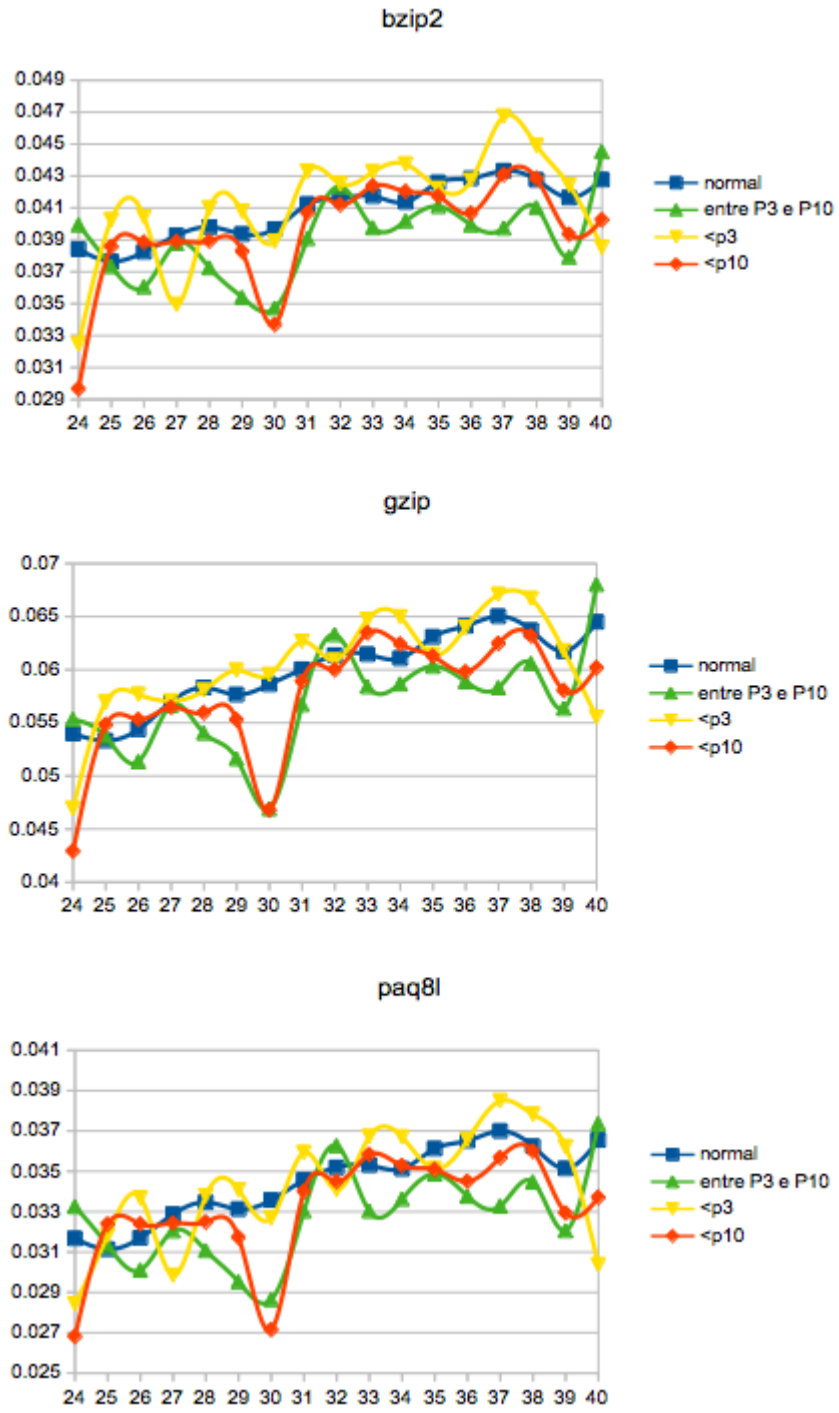


Figura 4.6: Relação entre o peso do bebé à nascença e a taxa de compressão dos traçados ao longo da gravidez usando os 3 compressores, onde p3 indica o percentil 3 e p10 o percentil 10.

4.2 Análise *Wavelets*

Em virtude de não termos notado uma clara vantagem no uso de um compressor em detrimento de outro na secção anterior, a compressão realizada nos testes desta secção foi feita pelo compressor paq8l.

Os primeiros testes realizados com a técnica das *wavelets* consistiu numa base de dados com 67 traçados do período anteparto, dos quais 48 normais, 10 patológicos e 9 suspeitos, de acordo com uma análise à posteriori por parte dos especialistas que avaliaram o bem estar do bebé à nascença através do pH do cordão umbilical. Para cada grupo foram calculados os rácios de compressão sem *wavelets* e com a aplicação das mesmas. Para isso usámos as *wavelets Haar, Daubechies e Coiflet*, aplicando sempre o *threshold* universal. Pela análise da Figura 4.7 notámos uma similaridade no comportamento das *wavelets Daubechies e Coiflet*, ao contrário da *wavelet Haar*. Neste último caso, encontrámos acentuações nas diferenças entre os rácios de compressão essencialmente para os traçados considerados normais, como se pode ver na Figura 4.8.

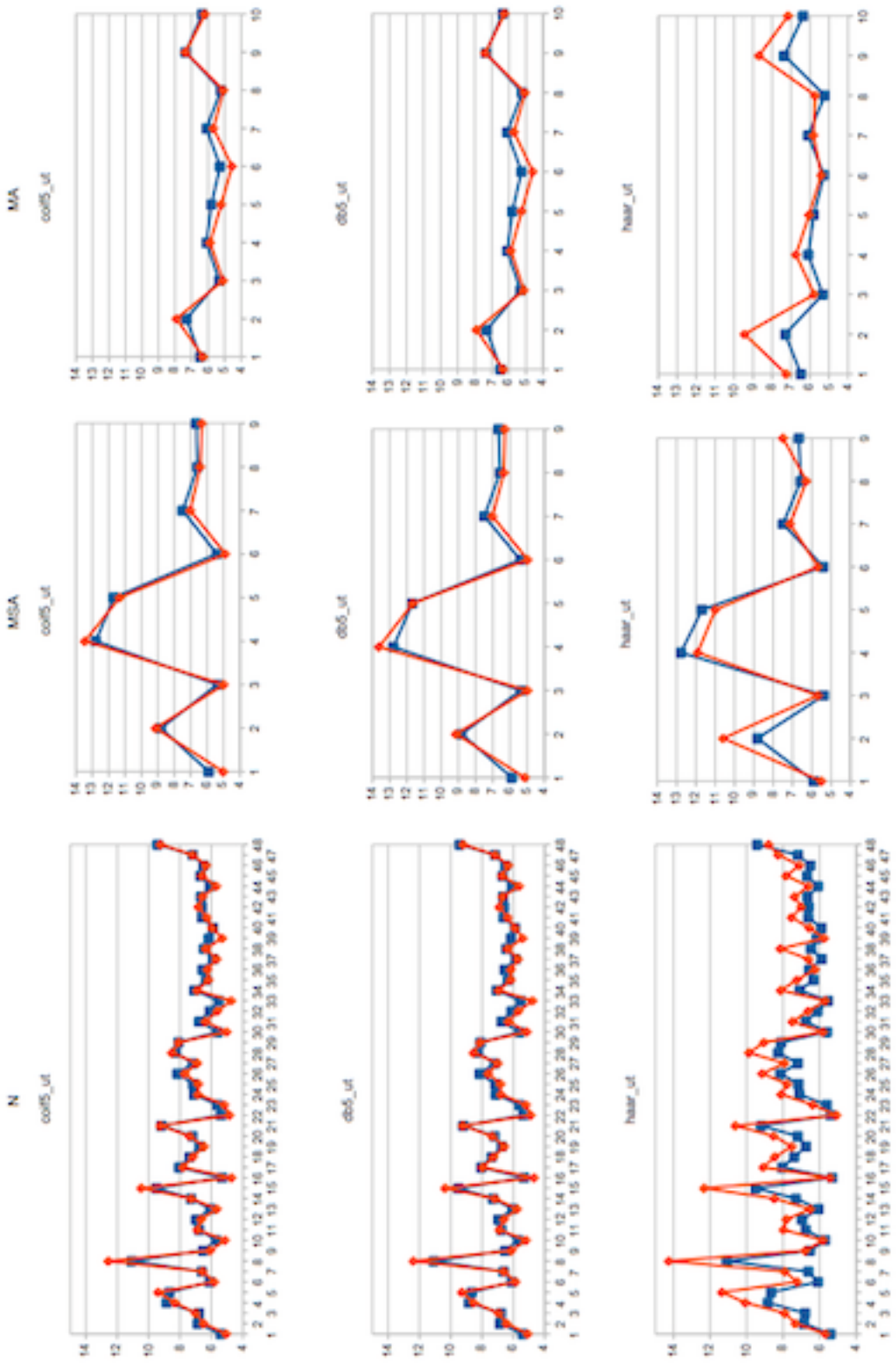


Figura 4.7: Gráficos de comparação entre o rácio de compressão sem utilização de wavelets e com a utilização das wavelets Daubechies, Coiflet e Haar para traçados normais (N), suspeitos (MSA) e patológicos (NA), onde os traçados azuis representam o rácio de compressão sem wavelet e os vermelhos com o wavelet respetivo

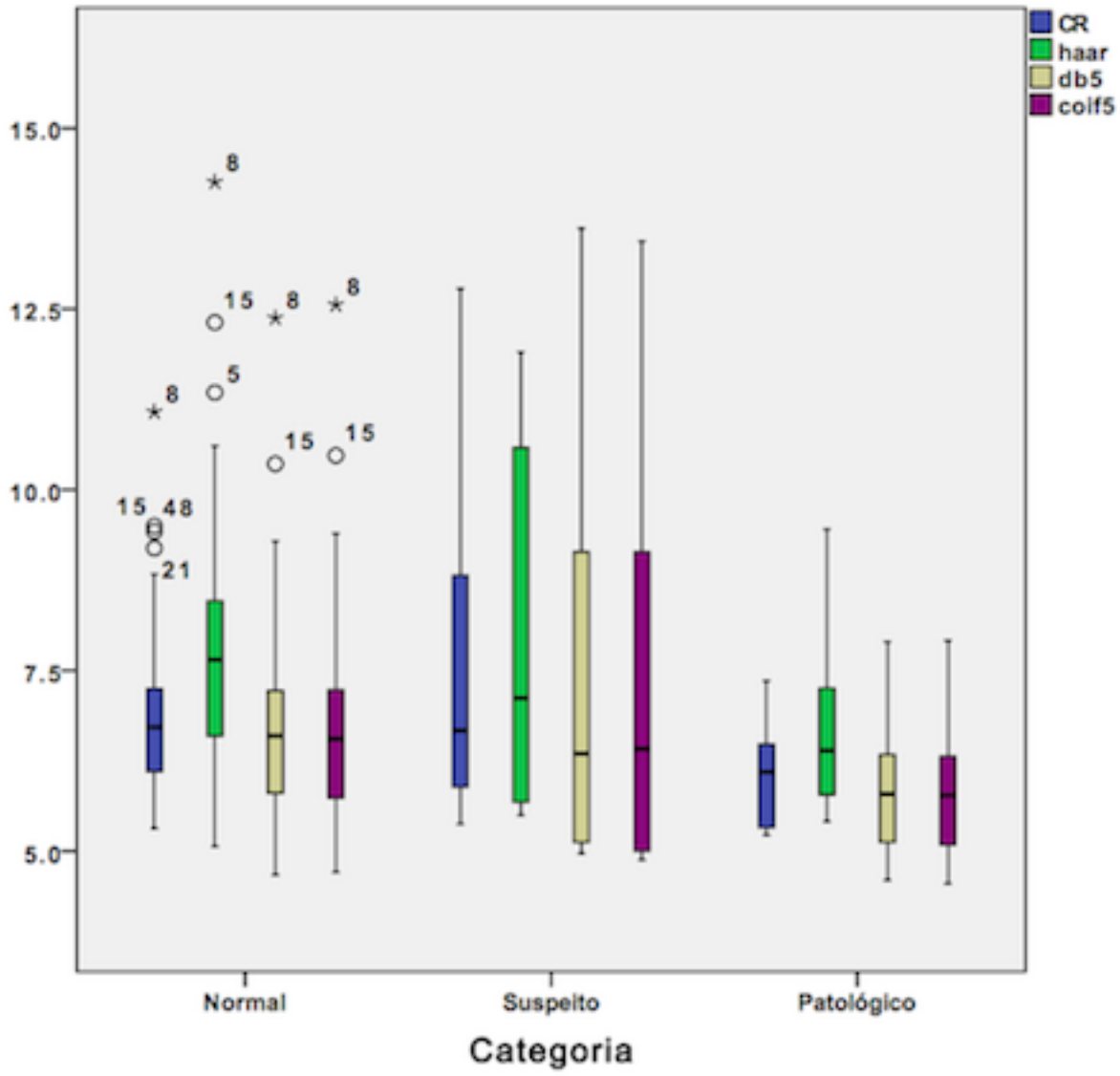


Figura 4.8: Gráficos de caixa com a comparação entre o rácio de compressão das wavelets Daubechies, Coiflet e Haar para traçados normais (N), suspeitos (MSA) e patológicos (NA)

No sentido de avaliar melhor o desempenho da *wavelet Haar*, fizemos o mesmo teste mas para um conjunto de dados maior consistindo em 5877 traçados de categoria normal, 120 de categoria suspeito e 34 da categoria patológico. O que observámos (Figura 4.9) foi um comportamento muito semelhante não só entre as diferentes categorias como também na utilização ou não das *wavelets*. Foi também testada a substituição do *threshold* universal por um *threshold* constante de valor absoluto

superior, o que resultou num maior número de coeficientes de detalhe *wavelet* anulados. Essa alteração, embora notada por um rácio de compressão maior, não trouxe qualquer benefício.

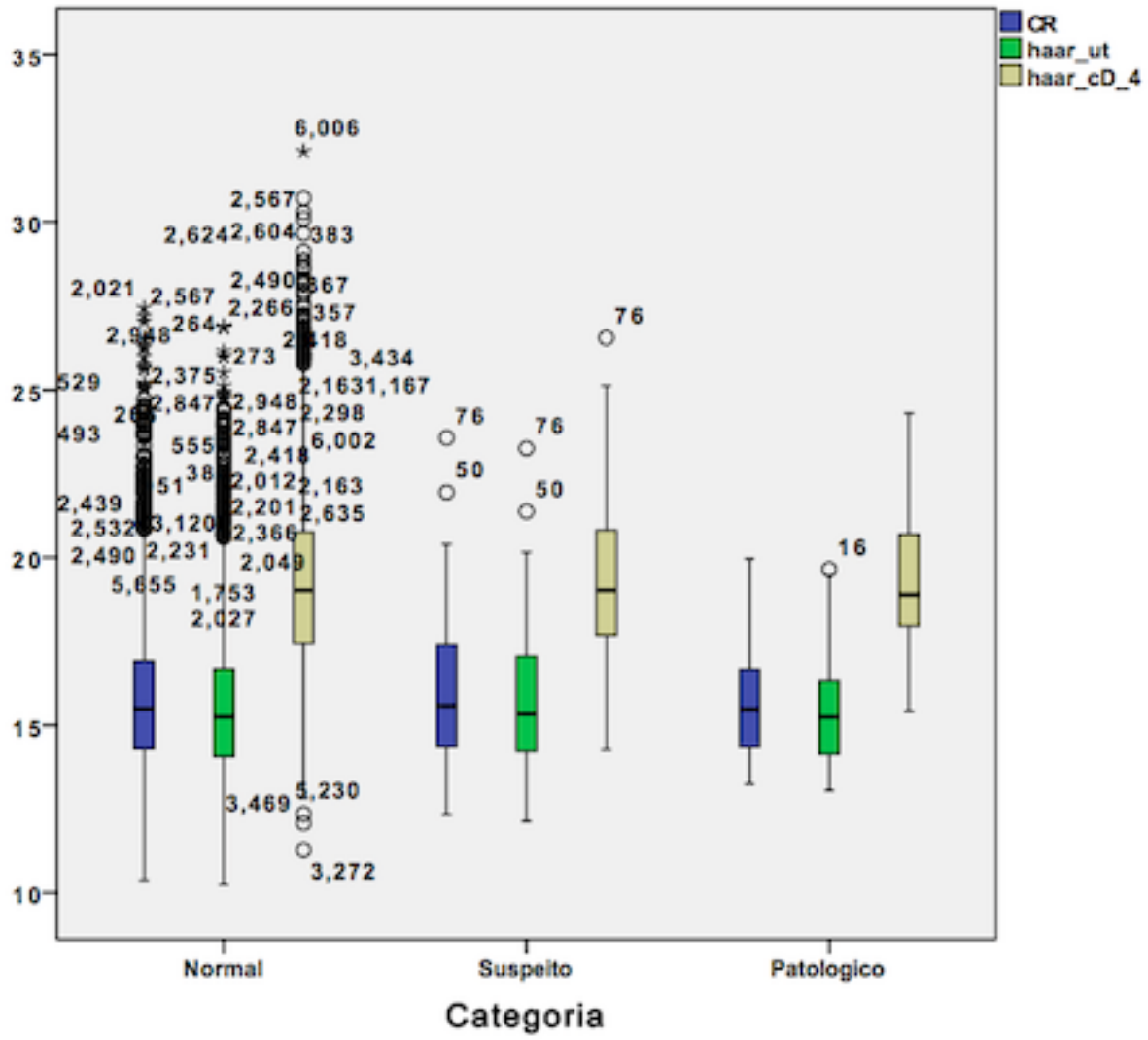


Figura 4.9: Gráficos de caixa com a comparação entre as wavelets Daubechies, Coiflet e Haar para as categorias normal, suspeito e patológico

Do mesmo conjunto de dados foram selecionados todos os traçados referentes à semana 30 da gravidez. O objeto de estudo foi tentar perceber se era possível prever a prematuridade de um feto através da variabilidade curta ou compressão e se a utilização de *wavelets* trazia algum benefício. De um total de 368 traçados definimos prematuro o bebé que nascera antes das semanas 37 (54 prematuros), 34 (19 prematuros) e 33 (14 prematuros). Constatámos que nas três categorias os resultados foram algo similares.

De notar uma melhoria na previsão sempre que a *wavelet Daubechies* foi aplicada, em comparação com a compressão usual e resultados positivos para a variabilidade curta nas semanas 33 e 34.

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	Comp_ratio	.173	.079	4.793	1	.029	1.189
	Constante	-7.728	2.142	13.018	1	.000	.000

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	db15_CR	.197	.073	7.316	1	.007	1.218
	Constante	-8.302	1.985	17.495	1	.000	.000

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	STV	.090	.035	6.456	1	.011	1.094
	Constante	-8.202	2.064	15.787	1	.000	.000

Figura 4.10: Análise do rácio de compressão (Comp_ratio), wavelet Daubechies (db15_CR) e variabilidade curta (STV) na predição de prematuridade (< 33 semanas)

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	Comp_ratio	.166	.069	5.773	1	.016	1.181
	Constante	-7.214	1.862	15.014	1	.000	.001

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	db15_CR	.192	.064	9.037	1	.003	1.211
	Constante	-7.819	1.722	20.610	1	.000	.000

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	STV	.102	.031	10.479	1	.001	1.107
	Constante	-8.559	1.843	21.566	1	.000	.000

Figura 4.11: Análise do rácio de compressão (Comp_ratio), wavelet Daubechies (db15_CR) e variabilidade curta (STV) na predição de prematuridade (< 34 semanas)

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	Comp_ratio	.082	.045	3.280	1	.070	1.085
	Constante	-3.840	1.171	10.757	1	.001	.021

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	db15_CR	.088	.040	4.852	1	.028	1.092
	Constante	-3.947	1.020	14.969	1	.000	.019

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	STV	.029	.019	2.322	1	.128	1.029
	Constante	-3.302	1.040	10.082	1	.001	.037

Figura 4.12: Análise do rácio de compressão (Comp_ratio), wavelet Daubechies (db15_CR) e variabilidade curta (STV) na predição de prematuridade (< 37 semanas)

Ainda do mesmo *dataset* foram selecionados os traçados referentes aos cardiotocogramas realizados durante a semana 37 da gravidez. Agora, em vez da prematuridade,

foi o sexo do bebé que tentámos prever usando as variabilidades curta e longa do traçado, o pH, o peso do bebé à nascença, a compressão com e sem a aplicação das *wavelets*. Este conjunto de dados é formado por 3441 traçados, dos quais 1767 são do sexo masculino e 1674 do sexo feminino e as funções *wavelets* usadas foram a *Haar e Daubechies* com *threshold* universal. Concluimos através da Figura 4.13, que apenas o peso e a variabilidade curta (STV) eram bons preditores do sexo do bebé. Através de um teste ajustado, foi colocado de parte a hipótese de dependência entre as variáveis STV e peso (Figura 4.14).

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	CR	-.018	.011	2.583	1	.108	.982
	Constante	.357	.258	1.914	1	.167	1.428

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	haar_CR	-.017	.011	2.413	1	.120	.983
	Constante	.344	.258	1.770	1	.183	1.410

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	db15_CR	-.562	.651	.744	1	.388	.570
	Constante	1.866	2.226	.703	1	.402	6.462

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	pH	2.361	1.219	3.755	1	.053	10.606
	Constante	-17.301	8.786	3.877	1	.049	.000

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	LTV	.005	.006	.539	1	.463	1.005
	Constante	-.067	.038	3.047	1	.081	.935

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	STV	.017	.004	16.432	1	.000	1.017
	Constante	-.843	.198	18.193	1	.000	.430

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	Peso	.000	.000	23.555	1	.000	1.000
	Constante	1.272	.275	21.351	1	.000	3.568

Figura 4.13: Análise do rácio de compressão (Comp_ratio), wavelet Haar (Haar_CR), wavelet Daubechies (db15_CR), pH do cordão umbilical, variabilidade longa (LTV) e variabilidade curta (STV) do traçado e peso à nascença na predição do sexo à nascença

Variáveis na equação

		B	S.E.	Wald	df	Sig.	Exp(B)
Etapa 1[a]	STV	.017	.004	16.483	1	.000	1.017
	Peso	.000	.000	23.607	1	.000	1.000
	Constante	.483	.337	2.062	1	.151	1.621

Figura 4.14: Resultados do ajustamento das variáveis STV e peso para a previsão do sexo do bebé

Por fim, foram analisados traçados obtidos no período intraparto, isto é, durante o parto. De um conjunto de 148 traçados, 142 foram considerados normais por parte dos especialistas, enquanto 6 foram considerados patológicos. O objetivo seria comparar cada um dos grupos através do rácio de compressão juntamente com a aplicação das *wavelets Haar* e *Daubechies* usando *threshold* universal. Como se pode ver pelo diagrama de caixas da Figura 4.15, o uso de wavelets não trouxe vantagem até porque não existe distinção entre os traçados normais e os patológicos.

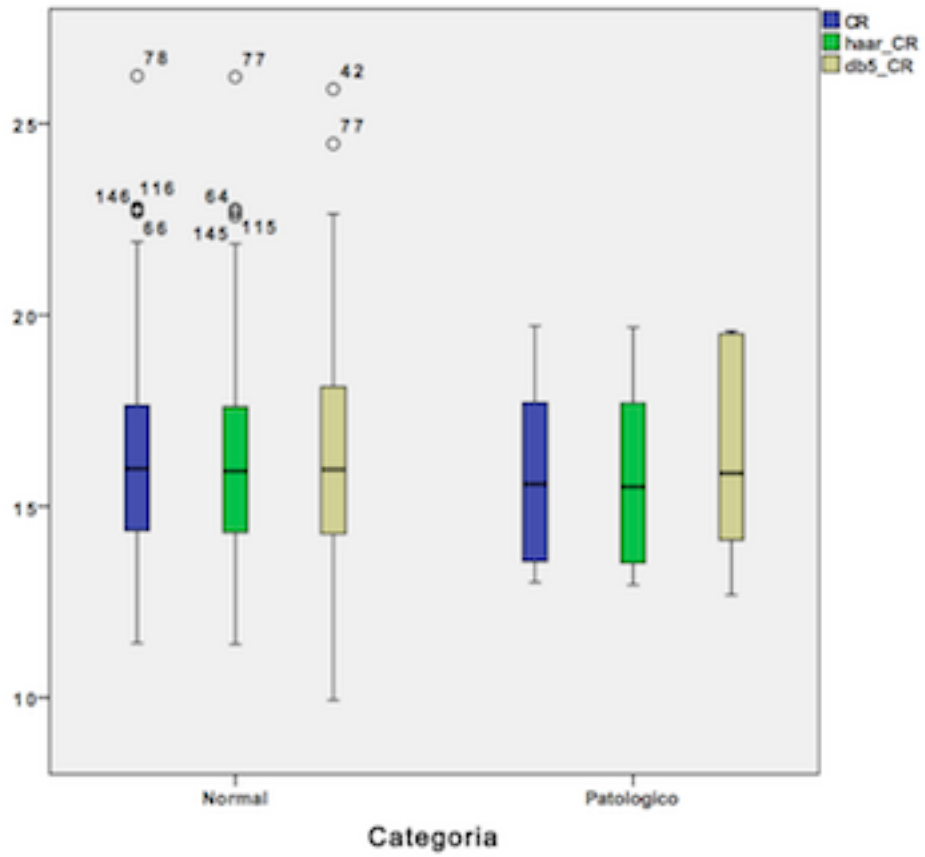


Figura 4.15: Gráfico de caixa de traçados obtidos no período intraparto. Rácios de compressão sem (CR) e com o uso de wavelets (Haar e Daubechies usando threshold universal)

4.3 Análise CompLearn

O objetivo desta secção é avaliar o comportamento do CompLearn na análise dos traçados de batimento cardíaco fetal. De um *dataset* com um total de 14812 entradas, foram selecionados os traçados referentes à semana 37 da gravidez no sentido de garantir que todos os fetos estavam nas mesmas condições, isto é, 3522 traçados. Em virtude da árvore resultante do algoritmo do CompLearn requerer tempo exponencial em relação ao número de entradas, tivemos de restringir este número nos testes que

se seguem. Em primeiro lugar testámos se o algoritmo seria capaz de distinguir os traçados cujo pH do cordão umbilical à nascença eram muito baixos dos muito altos. Posteriormente foi feito o mesmo mas para a variabilidade curta.

4.3.1 CompLearn vs pH

Foram selecionados 19 traçados cujo pH do bebé à nascença era inferior a 7 e outros tantos com pH superior a 7.2. O objetivo era percebe se o CompLearn tinha a capacidade de classificar os 38 traçados nestes dois grupos.

Como se vê pela Figura 4.16, a compressão não está relacionada com o pH do cordão umbilical medido à nascença.

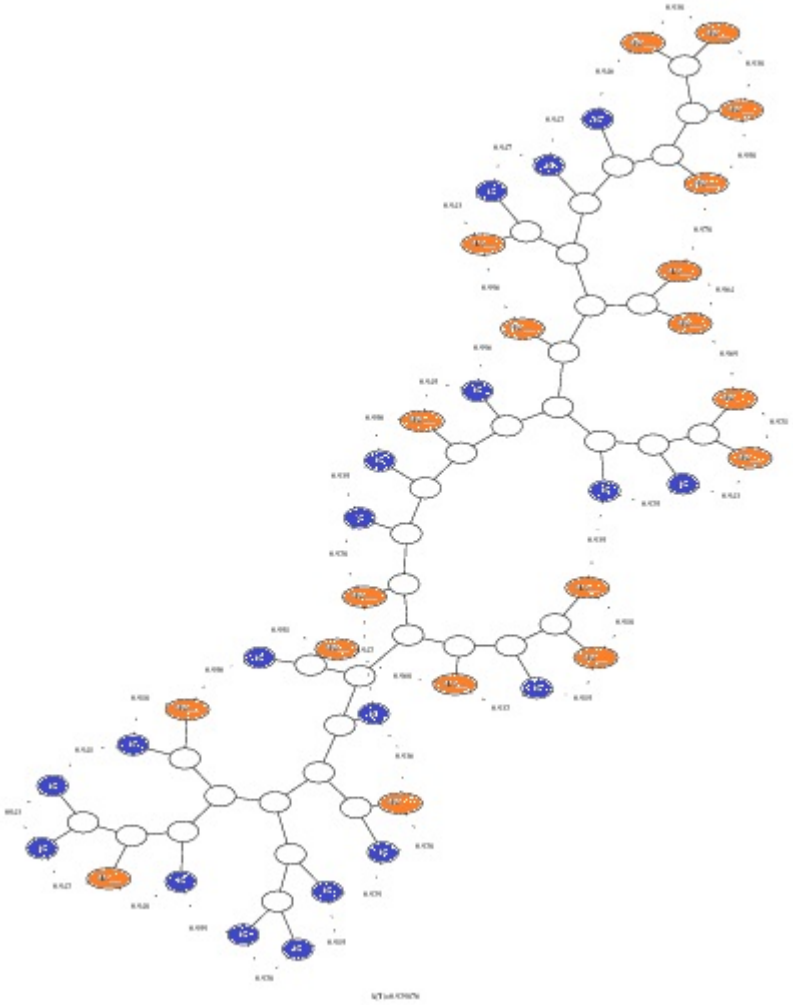


Figura 4.16: Árvore de classificação do Complearn para 38 traçados onde 19 são de bebés cujo pH do cordão umbilical à nascença era inferior a 2, e outros 19 superior a 2.2

4.3.2 CompLearn vs Variabilidade Curta

Foram selecionados 40 traçados, metade dos quais com variabilidade curta inferior a 20 enquanto na outra metade a variabilidade curta é superior a 70.

Como se pode ver na Figura 4.17, o CompLearn facilmente distingue os dois grupos.

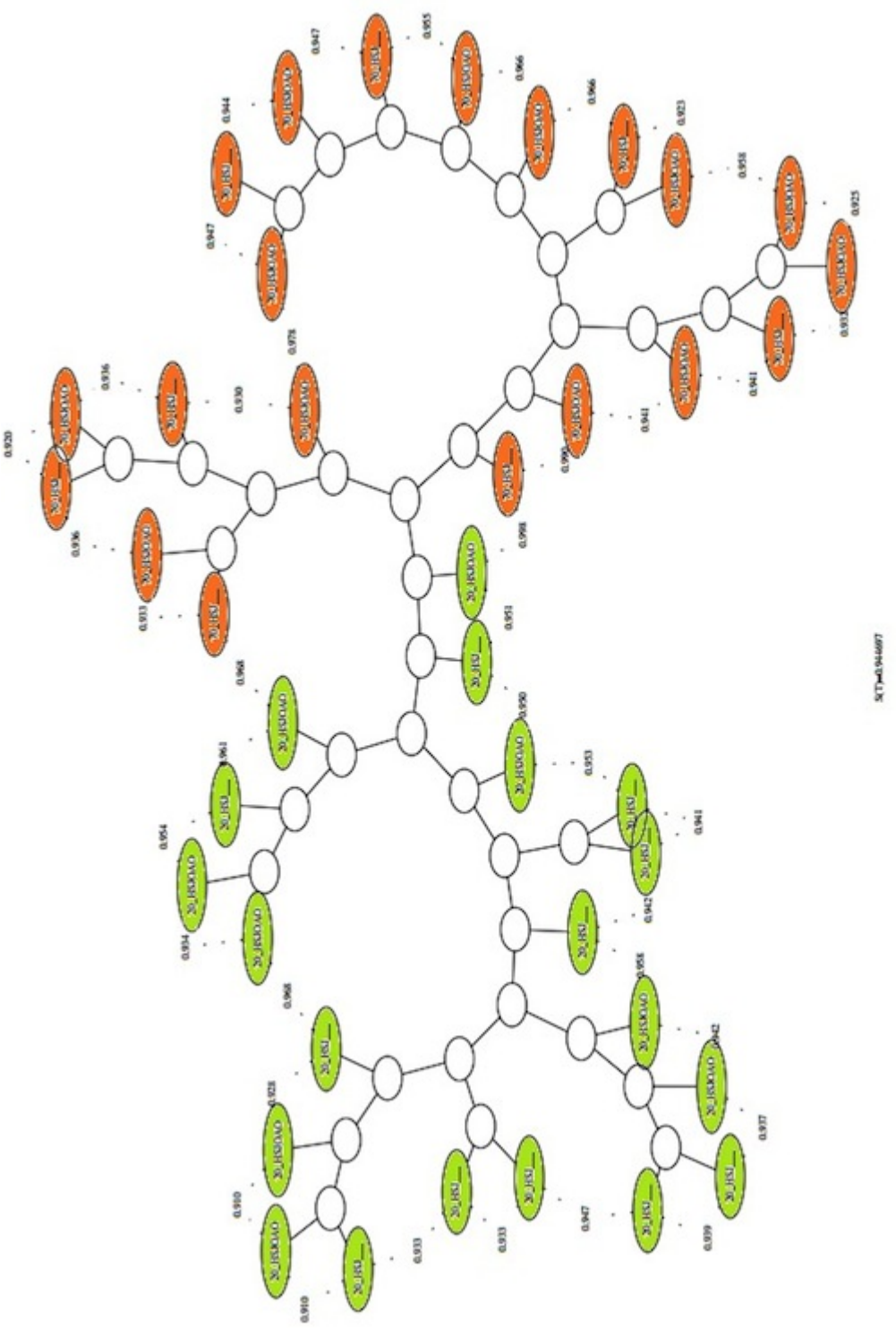


Figura 4.17: Árvore criada pelo CompLearn para traçados com diferentes variabilidades curtas (20 a verde e 70 a laranja)

No entanto, se adicionarmos 20 traçados cuja variabilidade curta ronda os 50, o algoritmo já não tem resultados tão promissores, mas ainda assim algo satisfatórios (Figura 4.18).

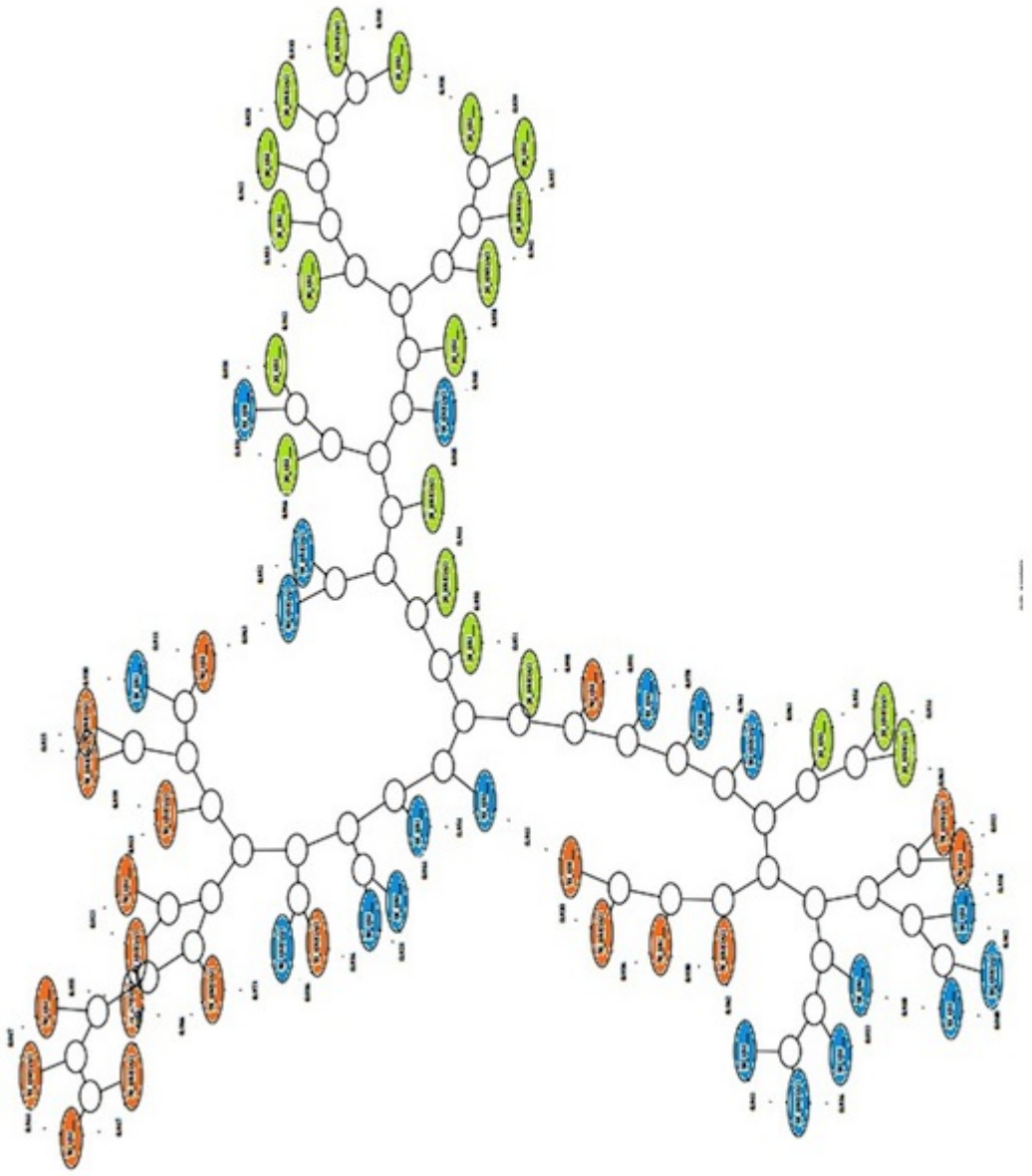


Figura 4.18: Árvore criada pelo Complearn para traçados com diferentes variabilidades curtas (20 a verde, 50 a azul, 70 a laranja)

Podemos assim concluir que de certa forma a compressão está relacionada com a variabilidade curta de um traçado.

Capítulo 5

Conclusão

Os resultados obtidos na dissertação de doutoramento [30] foram encorajadores e desta forma, nesta dissertação, pretendeu-se avaliar o interesse e utilidade de compressores com perdas (*wavelets*) na deteção de anomalias usando a compressão. A questão de investigação fundamental desta dissertação era averiguar se as perdas desses compressores teriam correspondência com o ruído inerente no sinal fisiológico e desta forma o método de classificação por compressão seria mais fiável.

Numa primeira fase do trabalho usámos compressores sem perdas para determinar a evolução dos fetos ao longo do tempo (semanas) a nível de complexidade/compressão dos traçados de batimento cardíaco fetal. Para tal, usamos três compressores diferentes e todos apresentaram a mesma dinâmica a menos de diferenças de escala. Efetuamos a mesma análise para tentar detetar diferenças entre sexos, no entanto a dinâmica parece ter um comportamento análogo para ambos os sexos. Depois estudamos a possibilidade de existirem diferenças entre os fetos com um bom vs mau *outcome* (pH do cordão umbilical) e, para este *dataset*, verficou-se um maior poder de compressão para os traçados patológicos ao longo de quase todas as semanas de gestação. Relativamente aos grupos de fetos pré-termo (habitualmente definido como nascer com 36 ou menos semanas) e termo normal, detetámos neste conjunto de dados, diferenças na dinâmica ao longo das semanas de gestação, o que significa que a compressão pode ser um bom indicador para esta variável. Por fim, relativamente às diferenças de peso (percentil 3, percentil 10, entre o percentil 3 e o 10, e os normais) foram detetadas diferenças na dinâmica dos vários grupos o que mais uma vez nos leva a acreditar que a compressão pode ser um bom preditor desta variável. De forma a suportar estes resultados, achamos relevante validar de forma mais rigorosa estes resultados aplicando as mesmas técnicas num conjunto de dados maior e usando metodologia

estatística para a comparação dos grupos.

De seguida usámos funções *wavelet* (que podem ser vistas como compressores com perdas) para comparar a sua eficácia neste tipo de abordagem relativamente aos compressores sem perdas, mais uma vez na analise de batimento cardíaco fetal. O *dataset* que usámos foi construído em [31] e pretendia-se agrupar os traçados em normal, suspeito e patológico. Os resultados pareciam prometedores para o caso da *wavelet haar*, no entanto após usarmos esta mesma *wavelet* num *dataset* de maior dimensão tal não foi o caso. Uma possível explicação para estes resultados pode ser baseada no facto de os 3 *clusters* terem sido definidos com base no pH do cordão umbilical que no primeiro estudo foi rigorosamente recolhido e no *dataset* com mais dados não, pois foram obtidos diretamente do sistema de informação do hospital. Estudámos também a utilidade das *wavelets* na predição de fetos prematuros com base nos traçados à semana 30. Neste caso, sabíamos que a compressão sem perdas o conseguia fazer com significado estatístico, mas com base nas *wavelets* conseguiu-se melhorar a qualidade da predição relativamente aos compressores sem perdas. Como trabalho futuro pretendemos avaliar quais as funções *wavelet* que melhor se adaptam a este tipo de dados. Será também interessante estudar os métodos de remoção de ruído (*denoising*), i.e., *threshold* e mais uma vez ajustar ao tipo de dados em causa.

Relativamente ao método *CompLearn*, selecionámos um conjunto de traçados da semana 37 e tentámos estudar qual a característica dos traçados que estava a ser explorada pelo método. Concluímos que a variabilidade curta dos traçados é o principal fator no *clustering* usando o *CompLearn*. Para trabalho futuro pretendemos criar uma implementação do algoritmo *CompLearn* com base nas *wavelets*. Sugerimos ainda tentar aplicar estes métodos a outros sinais fisiológicos e outros métodos de *data mining* clásicos em substituição da 2ª fase do *CompLearn* de forma a tornar o método mais célere.

Bibliografia

[1] Rudi Cilibrasi. Statistical inference through data compression. *ILLC Dissertation Series DS-2007-01*, 2007.

[2] F. Qiao. Introduction to wavelets - a tutorial. *Texas Southern University*, Jan 2005.

[3] Trisha Geenhalgh Paul E Plsek. The challenge of complexity in health care. *323:625-628*, BMJ 2001.

[4] A. Costa-Pereira J. Bernardes. Admission cardiotocography. *361(9370)pp.1741*, Lancet, 2003.

[5] Vitányi L. Antunes C. Santos, J. Bernardes. Clustering fetal heart rate tracings by compression. *Special Track on Data Mining, the 19th IEEE International Symposium on Computer-Based Medical Systems*, 2006.

[6] P. Vitányi R. Cilibrasi. Clustering by compression. *IEEE Transactions on Information Theory*, 2005, 51(4) pp.1523-45.

[7] M. Li P.M.B. Vitányi C.H. Bennett, P. Gács and W. Zurek. Information distance. *IEEE Transactions on Information Theory, 44(4):1407-1424*, 1998.

[8] X. Li B. Ma M. Li, X. Chen and P.M.B. Vitányi. The similarity metric. *Proc. 14th ACMSIAM Symposium on Discrete Algorithms*, 2003.

[9] S.N. Talbar S.O. Rajankar. An optimized transform for ecg signal compression. *ACEEE Int. J. on Signal and Image Processing*, Vol.1, No.5, Dec 2010.

[10] Mohammed Abo-Zahhad. Ecg signal compression using discrete wavelet transform.

[11] Carl Taswell. The what, how, and why of wavelet shrinkage denoising. *Computational Toolsmiths, Stanford, CA 94309-9925*, Jan 1999.

[12] Amina Khatun M. Mozammel Hoque Chowdhury. Image compression using discrete wavelet transform. *IJCSI International Journal of Computer Science Issues*, Vol.9, No.4, Jul 2012.

[13] Luiz Antonio Lopez. Transformadas de wavelet e lógica fuzzy na inspeção por eddy-current em tubos de geradores de vapor de centrais nucleares. *IPEN, Autarquia Associada à Universidade de São Paulo*, 2002.

[14] G. Kaiser. A friendly guide to wavelets. pp. 44-45.

[15] W. Press et al. Numerical recipes in fortran, cambridge university press, new york. pp. 498-499, 1992.

[16] Amara Grasp. An introdution to wavelets. *IEEE Computational Science and Engineering*, Vol.2, No.2, Summer 1995.

[17] G. P. Nason. Choice of the threshold parameter in wavelet function estimation. *Department of Mathematics, University of Bristol, Bristol BS8 1TW, UK*, 1995.

[18] Tim Park. An introdution to wavelets and wavelet de-noising. *ISBN: 978-0-9891305-4-7*, 2014.

[19] C. Deuschle S. Schluter. Using wavelets for time series forecasting: Does it pay off? *Econstor*, 2010.

[20] Y. Meyer. Wavelets: Algorithms and applications. *Society for Industrial and Aplied Mathematics, Philadelphia*, pp. 13-31, 101-105, 1995.

[21] W. Dally. Wavelet compression on imagine. *Stanford University*, Apr 2002.

[22] I. Daubechies. Ten lectures on wavelets. *SIAM*, 1992.

[23] Y. Meyer. Wavelets and operators. *IEEE Transactions on Acoustics, Speech and Signal Processing 34 (1986), 434-441.*

[24] B. Vidakovic. Basics of wavelet transforms. *Bayesian Statistics: Handouts, Handout 20, ISyE8843A.*

[25] C. Herley M. Vetterli. Wavelets and filter banks: Theory and design. *IEEE Transactions on Signal Processing*, 1992.

[26] Anjum Khan B. Ismail. Image de-noising with a new threshold value using wavelets. *Journal of Data Science 10(2012), 259-270.*

[27] T. P. Barnwell M. J. T. Smith. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. Pattn Anal. Mach. Intell 11, 674-693*, 1989.

[28] I. M. Johnstone D. L. Donoho. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, June 1992.

[29] Xu Zhenzhen Tam Ning. The application of wavelet threshold de-noising in mobile oxygen saturation monitoring software. *Lancaster University*, 2011.

[30] Teresa Henriques. *Assessing Complexity of Physiological Interactions*. PhD thesis.

[31] D. Ayres-de-Campos J. Bernardes H. Gonçalves, A. P. Rocha. Linear and nonlinear fetal heart rate analysis of normal and acidemic fetuses in the minutes preceding delivery. *Med Biol Eng Comput. 2006 Oct;44(10):847-55.*